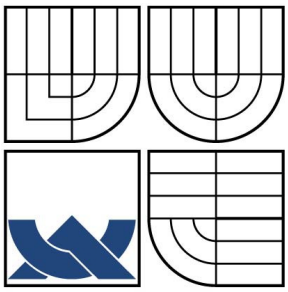
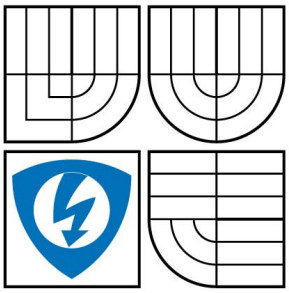


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

METODY VIZUÁLNÍHO VYLEPŠENÍ OBRAZU V INTERAKTIVNÍCH APLIKACÍCH DIGITÁLNÍHO TELEVIZNÍHO VYSÍLÁNÍ

Methods for image enhancement in interactive applications of digital telecasting

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — BC. VÍT ŠVANDA
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR ČÍKA
SUPERVISOR

BRNO 2008

ANOTACE

V této diplomové práci se zabýváme metodami minimalizující blokové artefakty v prostředí digitálního televizního pozemního vysílání. Tyto artefakty jsou způsobeny kompresemi založenými na kosínově transformaci (JPEG, MPEG2 I-Frames). Cílem této práce je proto vytvoření DVB-J aplikace jejíž hlavní funkcí je minimalizace blokových artefaktů u přirozených obrazů přenášených v DVB-T.

Nejdříve se proto zabýváme technologií digitálního vysílání DVB a platformu MHP, která zajišťuje funkci a provoz interaktivních aplikací. Dále definujeme rozdíly mezi jazykem Java a JavaTV. Popisujeme způsob, jakým se vyvíjí, simulují a spouští DVB-J aplikace. V další části této práce jsou rozebrány metody, kterými je možné detekovat a minimalizovat blokové artefakty. Následně je popsán návrh adaptivního filtru minimalizujícího blokové artefakty (dále jen MHP-MBA). Hlavním výsledkem celé práce je aplikace MHP-Deblocking, která v sobě implementuje jak vytvořený deblocking filtr MHP-MBA, tak filtr z video kodeku H.263. V závěru práce se věnujeme testování této aplikace na skutečném set-top-boxu v DVB-T vysílání.

Klíčová slova: MHP, DVB-J, Deblocking, DVB-T, Xlet

ABSTRACT

In this dissertation we deal with methods of minimizing block artefacts in digital terrestrial TV broadcasting. These artefacts are caused by compressions based on the cosine transformation (JPEG, MPEG2 I-Frames). The aim of this work is therefore to create a DVB-J application the main function of which would be to minimize block artefacts in natural images transmitted by DVB-T. That is why we first inquire into the technology of DVB digital transmission and MHP platform which provides the function and the running of interactive applications. Next, we define differences between Java and JavaTV languages and describe the way they develop, simulate and start DVB-J applications. In the following part of this work, we analyze methods that can be used to detect and minimize block artefacts. Further on, we describe the design of an adaptive filter which minimizes block artefacts (hereinafter just ´MHP-MBA´). The major result of the entire work is an MHP-Deblocking application that in itself implements a newly created MHP-MBA deblocking filter as well as a filter from video kodek H.263. In the final part we concern ourselves with testing this application on a genuine set-top-box in DVB-T broadcasting.

Keywords: MHP, DVB-J, Deblocking, DVB-T, Xlet

PROHLÁŠENÍ

Prohlašuji, že svůj semestrální projekt na téma "**Metody vizuálního vylepšení obrazu v interaktivních aplikacích**" jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne …… …….

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

Ad	Average different	Průměrná změna přechodů blokových artefaktů
AIT	Application Infomation Table	Tabulka obsahující informace o MHP aplikacích
DCT	Discrete Cosine Transform	Diskrétní kosínova transformace
DSM-CC	DownloadServerInitiate Messages Two-layer Carousels	Způsob přenosu interaktivních aplikací v DVB-T
DTS	Digital Theatre Systéme	Standard domácího kina
DVB	Digital Video Broadcasting	Digitální televizní vysílání
DVB-C	Digital Video Broadcasting – Cable	Digitální kabelová televize
DVB-H	Digital Video Broadcasting – Handy	Digitální televize do ruky
DVB-J	Digital Video Broadcasting – JAVA	Standart digitální televize s implementací JAVA standardu
DVB-S	Digital Video Broadcasting - Satellite	Digitální satelitní televize
DVB-T	Digital Video Broadcasting - Terrestrial	Digitální pozemská televize
DVD	Digital Video Disc	Optické medium
FIR	Finite Impulse Response	Filtr s konečnou impulsovou odezvou
GEM	Globally Executable MHP	Upravený standard MHP pro celosvětové použití
HTML	HyperText Markup Language	Internetoví protokol

I-Frame	Intra - Frame	Klíčový (úplný) snímek
P-Frame	Predicted – Frame	Předpovězený pomocný snímek
MHP	Multimedia Home Platform	Domácí multimediální platforma
MHP-MBA	Multimedia Home Platform – filtr Minimalizující Blokové Artefakty	Filtr minimalizující blokové artefakty určený pro interaktivní MHP aplikace
MPEG	Moving Picture Experts Group	Skupina zabývající se zpracováním video sekvencí
MSE	Mean Squared Error	Střední kvadratická chyba
PAL	Phase Alternating Line	Obrazový televizní standard
PSNR	Peak Signal-to-Noise Ratio	Špičkový odstup signál – šum

1 Úvod

Interaktivní aplikace v prostředí pozemního digitálního vysílání (DVB-T) využívají pro přenos přirozených obrazů formáty JPG, případně MPEG 2 I-snímky. Vysoký kompresní poměr obou těchto formátu je založen na kosínově transformaci a následné kvantizaci transformovaných koeficientů. Jedná se tedy o ztrátovou kompresi, která při málem datovém toku způsobuje výraznou degradaci obrazu. Visuálně rušivé jsou obzvláště blokové artefakty, které vznikají díky zpracovávání po blocích 8x8 pixelů. Tyto blokové artefakty mohou být úspěšně minimalizovány s využití speciálně navrhnutých filtrů, které jsou běžně využívány například na platformě PC.

Set-top-boxy které jsou v dnešní době na trhu, nemají žádné programové ani hardwarové vybavení, které by blokové artefakty potlačovalo. Cílem této práce je tak vytvořit software umožňující visuální vylepšení přirozených obrazů ve formátu JPG a MPEG2 I-snímky. Základem tohoto softwaru bude adaptivní filtr minimalizující blokové artefakty (dále jen MHP-MBA), jehož návrhem a detailní funkčnosti se budeme také zabývat. Vzhledem k omezenému výkonu dnešních set-top-boxů musí mít navržený filtr především malou výkonovou náročnost. Z tohoto důvodu bude pracovat pouze v prostorové oblasti. V závěru práce budeme funkčnost vytvořené aplikace ověřovat ve skutečném DVB-T vysílání.

2 Digital Video Broadcasting

Digital Video Broadcasting je evropský standard vyvíjený od roku 1991. Hlavním cílem standardu bylo sjednotit různá technická řešení při nevyhnutelném příchodu digitalizace televizního vysílání. To zabránilo vzniku více technologií řešící stejnou problematiku a následnému roztříštění trhu. První část standardu byla dokončena v lednu roku 1994 a v tom samém roce se k projektu přidává i Česká televize. V dnešní době má projekt DVB více jak 300 členů.

Myšlenka přejímání již existujících standardů je důvodem celosvětového úspěchu DVB. Implementace dobře známých standardů jako MPEG ulehčila a urychlila standardizační a legislativní začlenění v jednotlivých státech.

2.1 Rozdělení DVB:

- **DVB-S** -satelitní digitální vysílání
- **DVB-C** -kabelové digitální vysílání
- **DVB-T** -(*terestriální*) pozemní digitální vysílání

Rozdělení do tří formátů řeší problémy spojené s přenosovými podmínkami. Například satelitní přenos má malou úroveň užitečného signálu. Na druhou stranu disponuje širokým kmitočtovým pásmem. Naopak kabelový a pozemní přenos má k dispozici malou šířku kmitočtového pásma, ale s relativně silnou úrovní signálu.

Nejnovějším přírůstkem DVB je DVB-H (*Handy*). Tento formát je určen pro mobilní zařízení. Hlavní rozdíl je v úpravě způsobu posílání dat. Ty se na rozdíl od předchozích formátů neposílají spojitě, ale ve shluku impulsů (*burstech*). To má za následek až 90% úsporu energie mobilního zařízení. Dalším rozdílem oproti DVB-T je zmenšení rozlišení výsledného obrazu. Menší rozlišení totiž znamená i výrazné zmenšení potřebného datového toku [9].

2.2 Výhody DVB-T

Největší výhodou digitálního přenosu obrazu oproti analogovému je potřeba užšího kmitočtového pásma. U analogového vysílání je pro jeden televizní program potřebné pásmo o šířce 8MHz. Digitální vysílání dokáže ve stejném pásmu a srovnatelné kvalitě přenést až 5 programů. To je umožněno využitím kompresních

formátů. Ty redukují bitový tok na úroveň, kdy je obraz ještě dostatečně kvalitní. Tato úroveň je závislá pouze na kvalitách použitého kompresního formátu.

Digitální vysílání lze provozovat na stejném kmitočtu i na sousedních vysílačích. Zde dochází dokonce k překrývání signálů, což by u analogového vysílání způsobovalo vzájemné rušení. Proto se pro pokrytí takového území v analogovém přenose musí využívat ještě větší šířka kmitočtového pásma.

U digitálního přenosu odpadá problém vícečetných odrazů od okolních předmětů. Takový odražený signál dorazí k přijímači s fázovým posunem, který u analogového přenosu způsobuje tzv. „duchy“ v obraze. Odstranění tohoto neduhu umožní velkou mobilitu DVB-T zařízením. Ta se mohou pohybovat až 200 km/h a to bez ztráty kvality [9].

Vysílače šířící digitální přenos potřebují přibližně 10% energie k pokrytí stejného území jako analogové vysílače.

V bitovém toku je kromě televizního obrazu a zvuku možno posílat další multimediální služby, které se po připojení zpětného kanálu mohou stát plně interaktivní.

2.3 Nevýhody DVB-T

Je nutné vybudovat síť digitálního vysílání v kmitočtovém pásmu, ve kterém v současné době vysílají analogové vysílače. To je spojeno s náklady na obměnu vysílačů a nutností velkých přesunů v kmitočtovém spektru. Tyto přesuny budou postupně způsobovat zhoršování výkonu analogového vysílání a tím i kvalitu příjmu v jednotlivých lokalitách. Přechodné období, ve kterém budou vysílat analogové i digitální vysílače, si vynutí zvýšené náklady u provozovatelů vysílání.

Na straně uživatele je největší nevýhodou nutnost pořízení specializovaného zařízení, které bude přijímaný digitální signál převádět na analogový. Takové zařízení se jmenuje set-top-box a jeho cena se pohybuje od 1000 Kč [6].

2.4 Interaktivita

Interaktivní služby jsou jedním z největších lákadel, které digitální televize přináší. Samotný výraz „*interaktivita*“ znamená divákovu možnost změnit děj, probíhající na obrazovce. Pokud u současného analogového vysílání pomineme interaktivitu typu přepnutí programu, či vypnutí televize, zůstane nám jediná interaktivní služba „teletext“.

Ten je typickým představitelem interaktivní služby bez zpětného kanálu. Divák má možnost listovat pouze v informacích, které jsou obsaženy v paměti televize. Tyto informace se u analogové televize přenášejí v prvních čtyřech řádcích půlsnímkového zatemňovacího impulsu. Na stejném principu pracují i všechny interaktivní služby v DVB-T, které nemají zpětný kanál. Rozdíl je v přenosu dat. Ten neprobíhá během zatemňovacího impulsu, ale je neustále vysílán spolu s daty obrazu a zvuku. Teletext má ve srovnání s MHP primitivní výběr grafických komponentů a jeho potenciál dalšího vývoje je již nulový. Ačkoliv standard DVB jeho přenos umožňuje, počítá se s jeho zánikem spolu s ukončením analogového vysílání.

Nástupcem teletextu bude Superteletext. Jeho princip je podobný. I zde se informace jednotlivých stránek přenášejí ve stále se opakující smyčce. Vizuální stránka Superteletextu je ovšem mnohem příjemnější, umožňuje implementaci obrázků, filmových ukázek, reklam atd.

2.4.1 Služby s úplnou interaktivitou

Dosáhnout plné interaktivity služeb, znamená potřebu obousměrné komunikace mezi uživatelem a poskytovatelem dané služby. V oblasti digitální televize existovalo několik řešení jak tohoto obousměrného přenosu dosáhnout. Nadějně vypadala technologie DVB-RCT, ta předpokládala umístění malého vysokofrekvenčního vysílače[1] ke každému uživateli. Tento vysílač by realizoval zpětné spojení s poskytovateli digitálního vysílání. Technologie předpokládala nesymetrickou vzájemnou komunikaci mezi uživatelem a poskytovatelem. Pro zpětný kanál by tak byla dostačující i relativně nízká přenosová rychlost malých vysílačů. Ve směru od poskytovatele mělo být (v okruhu do 3,5 km od vysílače) dosahováno přenosové rychlosti, až několika Mbit/s. Důvod, proč nedošlo k uvedení této technologie do praxe, je nízká kapacita multiplexů (kmitočtového pásma). U nás by tato kapacita mohla být dostačující po úplném ukončení analogového vysílání (2010 – 2011), ovšem v této době budou zpětné kanály zcela jistě realizované jinými technologiemi.

U dnešních set-top-boxů je pro tvorbu zpětného kanálu využíváno především technologie Ethernet a integrovaného telefonního modemu. Ethernet umožňuje začlenění set-top-boxů do lokální sítě a následného využití kteréhokoliv dalšího připojení k internetu (ADSL, ISDN, 802.11).

[1] s výkonem do 0,5W.

Uživatel, mající set-top-box se zpětným kanálem, bude moci například přijímat a odesílat emailové zprávy, prohlížet internetové stránky, provádět internetové nákupy, stahovat filmy, hrát počítačové hry atd. Všechny tyto příklady záměrně nevyužívají pro „cestu“ k set-top-boxu digitálního vysílání, oba směry komunikace jsou realizovány výhradně prostřednictvím zpětného kanálu. Pojem „zpětný kanál“, tak ztrácí svůj pravý význam, který vyjadřoval simplexní přenos. Výše uvedené schopnosti budou set-top-boxy získávat postupně s rozvojem jejich technologie. Ta se bude pravděpodobně čím dál více přibližovat osobním počítačům, až nakonec vytvoří plnohodnotné domácí multimediální zařízení.

3 Multimedia Home Platform

DVB-T umožňovalo vysílání různých multimediálních služeb, ale na úrovni uživatele bylo potřeba definovat standard, který by umožňoval spouštění těchto služeb, jejich kontrolu a správu. Proto vzniklo několik na sobě nezávislých platforem. Tyto platformy nebyly vzájemně zcela kompatibilní. To bránilo jejich většímu rozvoji a dále to způsobovalo tříštění trhu. Proto bylo rozhodnuto o vytvoření společné platformy MHP.

Aby se tato platforma stala skutečně společnou a globální, je její specifikaci možno zdarma stáhnout. To umožňuje výrobu přijímačů zatíženou pouze malými poplatky na ověřovací zkoušku a nákladů vyplývajících z DVB licence. Základní myšlenkou MHP je umožnit výrobci (*programátorovi*) co nejvíce svobody. Aplikace určené pro MHP jsou proto napsány v jazyce JAVA nebo HTML. Tím se stávají nezávislé na jediné hardwarové platformě či operačním systému.

Platformu MHP vyvíjí experti DVB. Je určena především pro digitální vysílání využívající standard DVB-T. Pro skutečné globální využití MHP bylo nutné postihnout i destinace jako je Japonsko, USA, kde se využívají jiné standardy digitálního vysílání než DVB. Proto byla vyvinuta specifikace GEM (*Globally Executable MHP*). Tato specifikace vychází z MHP, ale jsou z ní odstraněny specifikace standardu DVB. Tím je umožněna kompatibilita i s jinými standardy.

3.1 Interaktivní služby v MHP

Hlavní podstatou interaktivního televizního systému je schopnost spouštět aplikace, které byly staženy z vysílaného datového toku. Právě tato schopnost odlišuje interaktivní přijímač od pouhého digitálního převodníku.

Jednou ze základních myšlenek MHP je neexistence samostatné MHP aplikace. To znamená, že každá MHP aplikace je řízena nějakou službou. Jestliže divák využívá MHP aplikaci spojenou s určitým televizním kanálem a tento kanál přepne, dojde většinou k jejímu vypnutí. To vede k omezení aplikací, které mohou běžet neustále, bez závislosti na určitém kanálu (službě). Provozovatelé tak dokáží lépe zaručit, že aplikace budou správně pracovat. Tento přístup zároveň usnadňuje divákovi navigaci mezi aplikacemi. Dostupné budou pouze ty, které souvisí s aktuálně sledovaným programem (*např. hlasování*).

Ovšem pouhé vysílání souborů tvořící aplikaci není dostatečné. MHP přijímač musí zjistit, k jaké aplikaci dané soubory patří a jak s nimi má zacházet. MHP proto

definuje informační aplikační tabuli AIT. Ta přijímač informuje o aktuálně dostupných aplikacích. Tyto informace shromažďuje komponenta aplikační manažer v MHP přijímači. Kromě sledování dostupných služeb, je aplikační manažer odpovědný za monitorování aktuálních služeb, může aplikace spouštět či ukončovat.

MHP aplikace mohou být napsány buď v Javě, nebo HTML. Platforma MHP ovšem definuje řadu rozšíření a omezení ve schopnostech obou jazyků. Z tohoto důvodu nejsou aplikace MHP slučitelné se standardní Javou či HTML. Pro rozlišení se tedy MHP verze Java aplikací označuje DVB–J a HTML verze DVB-HTML [4], [5].

3.2 HTML aplikace

Přidáním HTML podpory do MHP bude umožněn přístup k internetovým stránkám. To je velké lákadlo, které ovšem přináší několik problémů.

Většina HTML stránek je vytvořena pro rozlišení větší než 1024x768, ale rozlišení běžné televize je pouze 720x576, což způsobí celkovou nepřehlednost. Další problém představuje ovládání, které je vytvořeno především pro počítačovou myš.

Východiskem bude vytvoření speciálních stránek určených výhradně pro MHP. Posledním problémem je samotná podpora HTML aplikací. Ta je částečně přítomna ve verzích MHP 1.0. x, ovšem plná podpora byla přidána až ve verzi MHP 1.1. Díky nedostatečnému rozšíření této verze probíhá rozšiřování HTML aplikací velmi pomalu.

3.3 Application Infomation Table (AIT)

Jak již bylo uvedeno, informuje AIT o dostupných aplikacích. To ovšem není jediná informace, kterou poskytuje, dále to jsou řídící kódy: AUTOSTAR, PRESENT,DESTROY, KILL, PREFETCH, REMOTE.

- Aplikace, která obsahuje kód AUTOSTART, bude spuštěna automaticky po načtení do přijímače. To umožňuje spustit aplikaci, která přímo souvisí s právě vysílaným programem.
- Pokud přijímač příjme aplikaci s řídícím kódem PRESENT**,** uloží ji do seznamu dostupných aplikací a uživatel si ji může případně spustit.
- Aplikace s řídícím kódem KILL či DESTROY jsou přijímačem vypnuty. Toho je možno využít například při skončení určitého programu, se kterým byla aplikace svázána. Rozdíl mezi kódem KILL a DESTROY je, že při KILL má

uživatel možnost v dané aplikaci pokračovat. Při kódu DESTROY je aplikace vždy uzavřena.

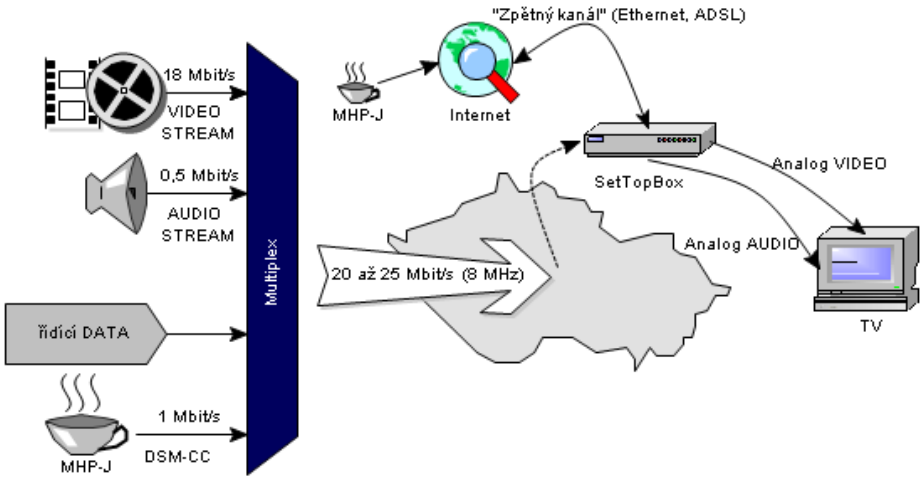
- Kód REMOTE přikazuje přijímači přejít na určitou službu za účelem spuštění aplikace poskytované danou službou.

Další důležitá informace udává pro jakou verzi MHP je aplikace určena. To dává přehrávači možnost rozhodnout, zda je danou aplikaci schopen správně interpretovat a tedy jestli ji zahrne do seznamu dostupných služeb či ne [3], [1].

3.4 Vysílání MHP aplikací

V dnešní době je nejčastější metodou vysílání aplikací DSM-CC objektový karusel. Tento přenos využívá zakódování aplikací do hlavního vysílacího toku spolu s video daty (MPEG2), audio daty (MPEG1 layer II) a řídícími daty (viz Obr. 3.1). Objektový karusel je určitý balík aplikací vysílaný v neustálé smyčce. Tento způsob přenosu přináší problém s množstvím dat přenášených v karuselu, toto množství je totiž přímo úměrné délce periody jedné smyčky, během které se zopakuje celý přenos dat. Délka periody určuje, jakou nejdelší dobu bude muset přijímač čekat, než bude moci začít stahovat aktuální data. V praxi se používá kompromis, který nalézá přijatelnou dobu nutnou k načtení celého karuselu.

Za rozšířeností této metody přenosu může především množství přijímačů, které ho podporují. To jsou všechny přijímače vybavené jakoukoli verzí standardu MHP. Navíc je objektový karusel pro přijímače bez zpětného kanálu jedinou možností jak přijímat MHP aplikace.



Obr. 3.1: Přenosový řetězec DVB-T a DSM-CC.

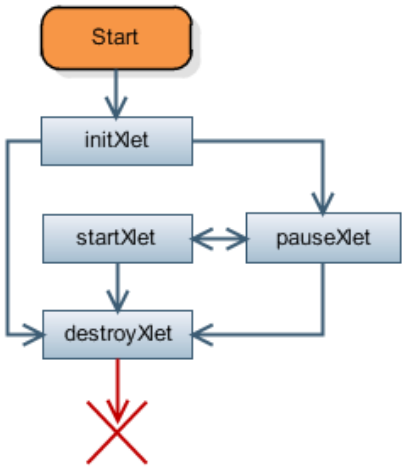
3.5 JavaTV

Jak bylo uvedeno, jsou aplikace určené pro MHP nejčastěji programovány v upravené verzi jazyka Java. Takovéto aplikace jsou označovány jako DVB-J a upravená Java nese označení JavaTV. Hlavním rozdílem oproti standardní Javě je implementace balíku *javax.tv* a jeho potomků. Zde se nachází základní třídy řídící běh DVB-J aplikací. Tyto aplikace jsou samozřejmě tvořeny i třídami, které nejsou obsaženy v *javax.tv*. Například zpětný kanál je programován výhradně třídami z java.net.

3.6 Xlet

U aplikace vytvořené ve standardní Javě se předpokládá, že v reálném čase bude prováděna pouze jediná aplikace. Digitální televize ovšem potřebuje spouštět více aplikací ve stejném čase. Vývojáři se tedy poohlédli po již existujícím řešení a to v podobě appletu.

Applet je spouštěn z webového prohlížeče a zavádí tzv. životní cyklus. Ten umožňuje aplikaci zastavit (pokud uživatel přepne danou webovou stránku) a ukončit ji (pokud je webová stránka s appltem uzavřena). Zároveň může být spuštěno několik appletů najednou. Applet byl proto přepracován pro podmínky digitální televize a výsledek se nazývá Xlet. I Xlet tedy obsahuje životní cyklus viz. Obr. 3.2, ale oproti tomu v appletu je složen ze stavů: „načtení“, „spuštění“, „pozastavení“ a „ukončení“. Stav „pozastaveno“ je využitelný, pokud je spuštěno více aplikací. V takovém případě bude viditelná pouze ta aplikace, se kterou právě pracujeme. Aplikace na pozadí přejdou do režimu „pozastaveno“ a uvolní zdroje (např. paměť) pro aktuální práce [1].



Obr. 3.2: Životní cyklus Xletu.

4 Měření vizuální kvality obrazu

Měření vizuální kvality obrazu můžeme dělit na subjektivní a objektivní. Pod pojmem subjektivní hodnocení si můžeme představit situaci, kdy daný obraz hodnotí několik nezávislých pozorovatelů. Je pravděpodobné, že každý z nich bude dané zkreslení obrazu vnímat rozdílně a s různou intenzitou.

Na rozdíl od subjektivního je objektivní hodnocení kvality obrazu založeno na empirických měřeních, jejichž výsledky jsou vždy absolutní. Základní myšlenkou je získání čísla, které bude reprezentovat kvalitu degradovaného obrazu v porovnání s originálním. Nejpoužívanějšími objektivními metodami jsou *MSE* a *PSNR*. Jedná se o metody, které nejsou založeny na fyziologických vlastnostech lidského zraku, ale na porovnání matematické podobnosti. Proto nemusí vysoká naměřená podobnost nutně znamenat i vysokou kvalitu. Na druhou stranu je výhodou těchto metod jednoduchost a výpočetní nenáročnost [10].

4.1 MSE

MSE (Mean Squared Error) měří tzv. reziduály mezi originálním a hodnoceným obrazem. Střední kvadratická chyba je definována jako:

$$MSE(x,y)=\frac{1}{m\cdot n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left[x(i,j)-y(i,j)\right]^2, \tag{4-1}$$

kde *x* je originální obraz, *y* je měřený obraz, *m* a *n* je počet řádků a sloupců, *i* a *j* jsou aktuální souřadnice v obraze.

4.2 PSNR

PSNR (Peak Signal-to-Noise Ratio) udává poměr mezi maximálním možnou energií signálu a energií střední kvadratické chyby hodnoceného obrazu. Z důvodu mnohdy velmi širokého dynamického rozsahu výsledků je PSNR udáván v decibelech. Špičkový odstup signál – šum je tedy definován jako:

$$PSNR(x,y)=10\cdot\log\left(\frac{MAX^2}{MSE(x,y)}\right), \tag{4-2}$$

kde *MAX* je maximální energie signálu, pro 8bitový signál je *MAX* = 255.

5 Dodatečné vylepšení kvality (post-processing)

Post-processingem rozumíme v oboru zpracování obrazu takovou úpravu, která má za cíl určitým způsobem vylepšit vlastnosti daného obrazu. Nejčastěji post-processingové filtry řeší blokové artefakty. Ty jsou způsobeny kompresními algoritmy, které využívají zpracování po blocích 8x8 a následnou kompresi založenou na DCT. Důvodem vzniku viditelných blokových artefaktů je nízký datový tok, který již neumožňuje dostatečné zachování detailních informací v porovnání s originálním blokem. To ve spojení s faktem, že obraz je zpracováván nezávisle po blocích, způsobuje viditelné hranice mezi jednotlivými bloky. Tyto hranice jsou tím výraznější, čím menší je datový tok.

Řešením je použití právě post-processingových filtrů, které v první fázi musí detekovat nežádoucí přechod a ve druhé fázi jej odstranit (minimalizovat). Post-processingových filtrů je celá řada, jejich vlastnosti se liší v závislosti na způsobu použití. Asi nejdůležitějším parametrem při výběru filtru je jeho výkonová náročnost. Ta je především kritická v případě použití u video sekvence. Vycházíme-li z normy PAL, je obnovovací frekvence 25 snímků za sekundu. Na provedení načtení, dekódování, post-processing a zobrazení tak máme maximálně 40ms. Pokud by byl tento čas překročen, dojde k zahození následujícího snímku. Může tak dojít k výraznému „zamrzávání“ obrazu a tudíž ke zhoršení celkové kvality. Výkonový prostředek je tak rozhodující ve volbě druhu post-processingového filtru.

Obraz můžeme obecně upravovat v prostorové a transformované oblasti. Ovšem filtrace v transformované oblasti (použití diskrétní kosínové, či vlnkové transformace) je výkonově velmi náročná. Její použití v set-top-boxu, který má procesor s taktem průměrně 200MHz, tedy není vhodné. Z tohoto důvodu se zaměříme výhradně na úpravy v prostorové oblasti [10].

5.1 Metody filtrace v prostorové oblasti

Metod filtrace v prostorové oblasti je velké množství. Liší se především svou adaptivitou, tedy způsoby jakými je upravována síla filtrace. Za další druh adaptivity můžeme považovat často používané více módové filtrování, což znamená použití dvou a více způsobů filtrace. Takovéto filtry mají výrazně větší efektivitu, dokáží totiž flexibilně reagovat na určité změny, či specifické oblasti v obraze. Adaptivita spočívá ve výběru právě jednoho způsobu filtrování pro danou situaci. Takovéto metody tedy musí sledovat určité parametry obrazu, jejichž změna dokáže indikovat vhodné použití daného módu.

5.2 Filtrování dolní propustí

Filtrování dolní propustí je nejjednodušší metodou. Spočívá ve filtrování vysokofrekvenčních informací obrazu. Často se používá v podobě matice 3x3, kde aktuální filtrovaný prvek je na pozici 1,1. Tato matice je naplněna aktuálním prvkem a jeho sousedními prvky. Výstupní aktuální prvek má hodnotu algebraického průměru této matice. Nevýhodou této metody je výrazná ztráta ostrosti obrazu viz Obr. 5.1 b). Pro omezení tohoto efektu je možné použít váhované konvoluční jádro (matici), která přidělí aktuálnímu prvku větší energie, čímž se vliv sousedních prvků zmenší viz Obr. 5.1 c) [11].

a)

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

b)

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{16} & \frac{1}{16} \\ \frac{1}{16} & \frac{8}{16} & \frac{1}{16} \\ \frac{1}{16} & \frac{1}{16} & \frac{1}{16} \end{bmatrix}$$

c)

Obr. 5.1: a) Bez filtrace, b) Filtrování dolní propustí,
c) Filtrování dolní propustí s upravenou váhovanou maticí

5.3 Mediánový filtr

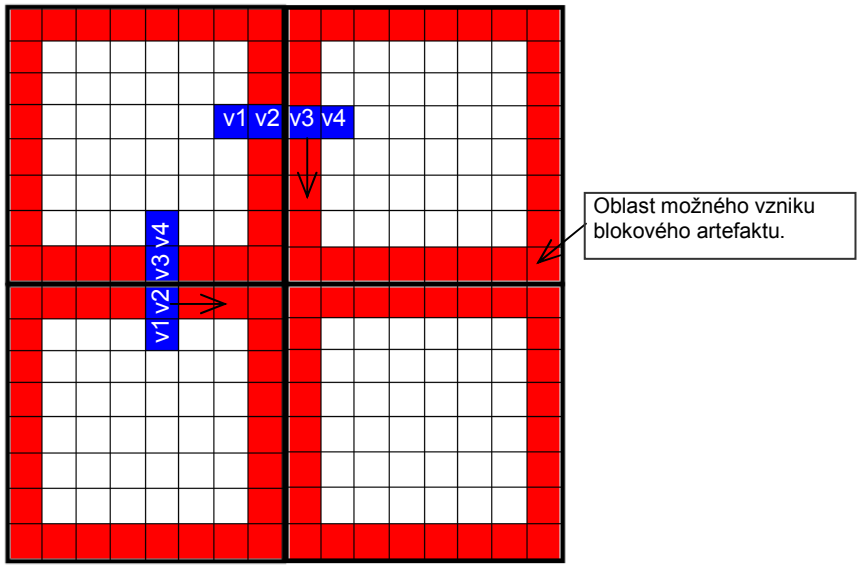
Mediánový filtr se chová jako dolnopropustný filtr typu FIR. Jeho činnost i výsledky jsou proto velmi podobné výše popsanému filtru dolní propust. Rozdílem je způsob výpočtu výstupního aktuálního prvku. U dolní propusti to byl algebraický průměr prvků matice, u mediánového filtru je vybrán prostřední prvek matice. Z tohoto způsobu vyplývá i další výhoda v podobě filtrace osamocených bodů (impulsního šumu). Nevýhodou je opět výrazná ztráta detailů v obraze [11].

5.4 Adaptivní filtry

Výše popsané filtry založené na dolní propusti dokáží výrazně snížit přechody blokových artefaktů. Bohužel nevýhodou, je již zmíněná ztráta detailů (ostrosti) obrazu. Hlavním důvodem této degradace je aplikace dolní propusti na všechny body obrazu bez rozlišení, zda-li se skutečně jedná o blokový artefakt. Pro zlepšení kvality je tedy nutné pracovat s adaptivními filtry.

Velmi jednoduchá a téměř vždy používaná metoda, která výrazně omezuje chybnou aplikaci filtru, vychází z poznatku, že blokové artefakty způsobené kompresí založené na DCT se mohou vyskytovat pouze v jasně definovaných intervalech. Tyto intervaly vyplývají z velikosti bloku (8x8) po kterých kodér zpracovává obraz. Jak již bylo popsáno výše, blokové artefakty vznikají právě na přechodech mezi těmito bloky, a proto je vhodné, aby post-processingové filtry byly aplikovány pouze na těchto hranicích, tedy pouze každých 8 pixelů (viz. Obr. 5.2).

V pokročilejších adaptivních filtrech jsou implementovány metody detekce hranic blokových artefaktů. I v těchto filtrech je ale tato metoda implementována. Využívá se její největší přednosti, kterou je výrazná úspora výkonu. Důvod této úspory je nasnadě, blokové artefakty jsou vyhledávány v jednom směru po jednotlivých prvcích a ve druhém směru po každém osmém prvku. Dochází tak k osminásobnému zvýšení efektivity filtru jak v horizontálním, tak ve vertikálním cyklu [12].



Obr. 5.2: Oblasti možného vzniku blokového artefaktu.

5.5 Adaptivní prahování

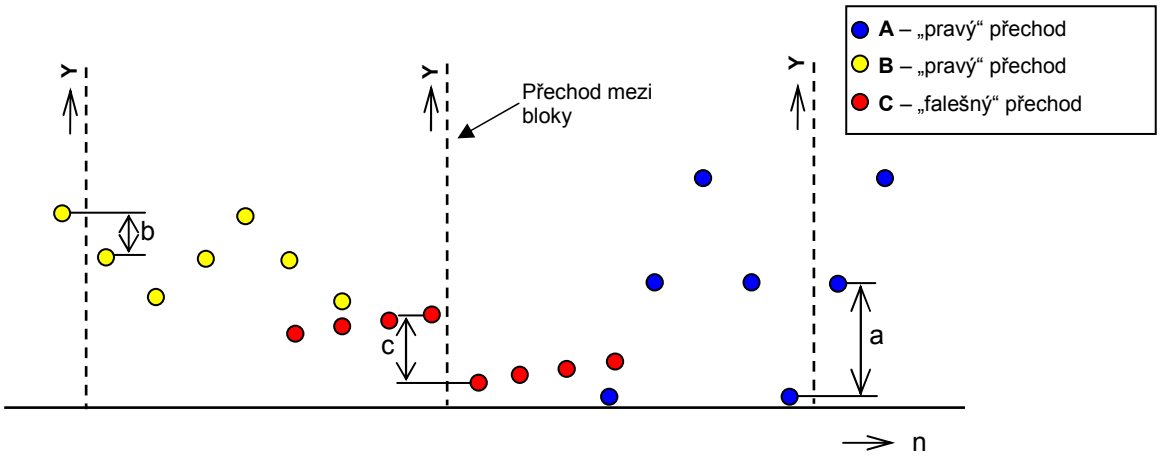
Aby byl post-processingový filtr skutečně efektivní, je nutné jeho filtraci aplikovat pokud možno pouze na visuálně „rušivou“ hranici způsobenou kompresí založenou na DCT. Algoritmus musí dokázat rozhodnout, nejen, zda-li se nachází v testované oblasti výrazný přechod, ale musí rozlišit, jestli se jedná o „falešný“ nebo o „pravý“ přechod, který tvoří detailní informaci původního nekomprimovaného obrazu.

Na Obr. 5.3 jsou zobrazeny tři průchody přes hranice bloků 8x8. Průběh „A“ představuje velmi výrazný „pravý“ přechod, který se pouze náhodou ocitl na hranici bloků. Kdyby došlo k jeho vyhlazení post-processingovým filtrem, snížila by se nežádoucím způsobem ostrost obrazu. Totéž platí i pro průběh „B“, ten také představuje „pravý“ přechod, ale s malou změnou. Ani v tomto případě není aplikace filtrace žádoucí. Poslední průběh „C“ představuje „falešný“ přechod způsobený ztrátovou kompresí.

Detekce „falešných“ přechodů je založena na poznatku, že můžeme vysledovat závislost mezi intenzitou „falešných“ přechodů a kvalitou (q) komprese tzn. na použitých kvantovacích tabulkách. Při určité kvalitě komprese tak mají „falešné“ vizuálně rušivé přechody velmi podobnou intenzitu přechodů mezi bloky. Vyhledávací algoritmus filtru tak nejsilněji reaguje na přechod právě s touto intenzitou. Pokud je detekován menší či větší přechod je mu přidělena váha, která s rostoucí vzdáleností od referenční intenzity úměrně klesá. Přidělená váha určuje sílu, s jakou bude daný přechod rozmazán.

Nevýhodou této metody je situace, kdy se objeví „pravý“ přechod s intenzitou změny stejnou, či podobnou intenzitě referenční. Dojde tak k výraznému potlačení tohoto přechodu a tím i k nežádoucí ztrátě detailů v obraze. Z globálního hlediska je ovšem tato situace velmi ojedinělá a tak tato metoda naopak nežádoucí ztrátu detailů v obraze výrazně snižuje.

Další nevýhodou je nutnost znalosti referenční intenzity, potažmo hodnoty kvality, se kterou byl daný obraz komprimován. U formátu typu JPG je možné tuto hodnotu odvodit z kvantizačních tabulek ukládaných v hlavičce daného souboru. Ze znalosti kvality komprese můžeme podle převodní funkce získat požadovanou referenční hodnotu.



Obr. 5.3: Průběhy na přechodech bloků.

5.6 Minimalizace přechodu – dolní propustí

Způsobů minimalizace detekovaných rušivých přechodů je několik. Nejjednodušší je aplikace 1D dolní propusti, ta je vidět na obrázku Obr. 5.2 a často se k tomu využívá vektor o velikosti 4, který protíná rušivý přechod. Z pixelu přechodu *v2* a mimohraničního pixelu *v1* se vypočítá aritmetický průměr, který se stane novým hraničním pixelem *v2*. Stejným postupem získáme i druhý hraniční bod, tedy opět aritmetickým průměrem tentokrát pixelů *v3* a *v4*. Tímto postupem dosáhneme menšího přechodového rozdílu mezi hraničními a prvními mimohraničními pixely. Pokud byl ovšem rozdíl levé a pravé strany přechodu příliš velký, nedosáhneme optimálního vyhlazení [12].

5.7 Minimalizace přechodu – linearizací

Účinnější metodou je linearizace přechodu. Ta na rozdíl od dolní propusti nevyužívá k výpočtu nového přechodu původní hraniční body, ale přechod je odvozen pouze z okolních bodů. Opět můžeme vyjít z vektoru délky 4 z Obr. 5.2. Nejdříve zjistíme dynamiku mezi krajními body. Dále z této dynamiky vypočítáme lineární změnu mezi jednotlivými přechody (*d*), což provedeme vydělením dynamiky počtem přechodů. Posledním krokem je výpočet hraničních bodů. Pixel *v2* je roven součtu offsetu *v1* a *d*. Pixel *v3* je roven součtu offsetu *v1* a dvojnásobku *d*. Ukázka linearizace přechodu délky 8 je na Obr. 5.4. Obecně platí pro vektor délky *n*:

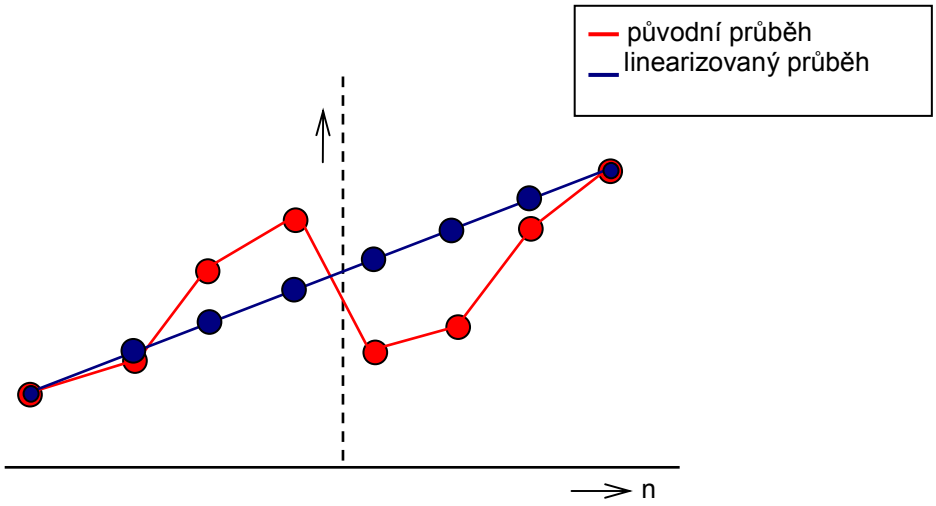
$$d = \frac{v_n - v_1}{n-1}, \tag{5-1}$$

$$v_i = v_1 + (i-1)\cdot d \qquad 1 \le i \le n, \tag{5-2}$$

kde *d* je lineární změna mezi přechody, *n* je délka vektoru a *i* je pořadí aktuálně počítaného prvku.

Linearizace přechodu s dlouhým vektorem, například *n* = 8 je velmi účinná metoda jak potlačit přechody v hladkých scénách obrazu. Zde je použití jiných metod málo efektivní a především ze subjektivního hlediska jsou tak přechody stále rušivé.

Naopak použití linearizace přechodu s vektorem délky 8 na oblasti s velkou dynamikou by způsobilo nežádoucí snížení detailů zapříčiněnou ztrátou informací o mimohraničních bodech (viz Obr. 5.4). Jako řešení je možné použít váhování vektoru, kdy budou jednotlivé linearizované body v závislosti na vzdálenosti od přechodu průměrovány s body původními [10], [11].



Obr. 5.4: Linearizace přechodu.

6 Filtr Minimalizující Blokové Artefakty (MHP-MBA)

Dílčím cílem této práce bylo vytvoření jednoduchého neiterativního filtru minimalizujícího blokové artefakty, který by pracoval v prostředí MHP. V první fázi bylo tedy nejdříve nutné vytvořit algoritmus samotného filtru, který by splňoval jak požadavky na dostatečné zlepšení objektivní i subjektivní kvality, tak i požadavky na dostatečnou rychlost potřebnou pro prostředí set-top-boxů.

Z počátku se jevil jako nejefektivnější způsob vyvíjet algoritmus přímo v jazyce JAVA (respektive v JavaTV). Kód by tak bylo možné ihned simulovat v prostředí IRT a následně by bylo nutné pouze testovat vytvořený program v prostředí Osmosis, potažmo na skutečném set-top-boxu. Ovšem samotný vývoj filtru v prostředí JavaTV se ukázal jako velmi problematický a zdlouhavý. Důvodů bylo hned několik, například simulace v prostředí IRT znamenala časté zamrzání zpuštěného programu a tím i nechtěných časových prodlev. Samotný vývoj v prostředí JavaTV je samozřejmě možný, ale ukázal se jako velmi nepohodlný. Problémem byly i velmi nízké možnosti následné analýzy výsledků. Řešením tedy bylo vyvíjet algoritmus filtru nejdříve ve vhodnějším prostředí jako je například MATLAB.

6.1 Vlastnosti filtru

Základní vlastnosti filtru bylo nutné podřídit výkonovému prostředku, na kterém bude provozován. Jak již bylo vysvětleno výše, tímto prostředkem je set-top-box s průměrným taktem procesoru 200MHz. Z tohoto důvodu nemohl filtr pracovat v kmitočtové oblast, ke které by bylo potřeba provádět výkonově náročnou diskrétní kosínovu, či vlnkovou transformaci.

Další omezení bylo nevyužití víceprůchodových (iterativních) procesů. Ty by například umožnily analýzu výskytu blokových artefaktů v celém zkoumaném obraze. Což by následně umožnilo jejich efektivnější minimalizaci. Nevýhodou víceprůchodového zpracování je ovšem nižší rychlost klesající úměrně s počtem průchodů.

V kapitole 5 jsou rozebrány vlastnosti filtrů založených na neadaptivní dolní propusti. Její výhodou je jednoduchost, rychlost, ale i výrazné potlačení blokových artefaktů. Naopak velkou nevýhodou je snížení vysokofrekvenčních informací i u užitečných dat, což způsobuje znatelné snížení detailů obrazu. Z tohoto důvodu bylo nezbytné vyvinout adaptivní filtr reagující na lokální vlastnosti obrazu (hranice blokových artefaktů).

Vytvořený filtr je adaptivní ve třech fázích:

- adaptivita na úrovni **detekce přechodů**,
- mód **hladkých oblastí**,
- adaptivita řízená **kvantovací tabulkou** souboru.

6.2 Adaptivita – detekce přechodů

U post-processingových filtrů je nejdůležitější adaptivní funkcí detekce existence blokových artefaktů a jejich síly. Z této znalosti může být odvozena váha, kterou je následně řízena síla filtrovaní daného přechodu.

Pro výpočet váhy aktuálního přechodu je nutné vypočítat *differenci* přechodu, která je úměrná jeho vážené dynamice a snížená o pravděpodobnost, že se nejedná o blokový artefakt.

Základem detekce přechodu je výpočet jeho dynamiky. Na Obr. 6.1 jsou zobrazeny dva průchody z nichž jeden je „pravý“ přechod a druhý přechod „falešný“ způsobený kompresí. Oba mají stejnou dynamiku na přechodu mezi bloky, ale rozdílnou dynamiku mezi mimohraničními body. Proto se od dynamiky přechodu *d1* odečítají absolutní rozdíly hraničních a prvními mimohraničními bodů. Pokud tento postup aplikujeme na „pravý“ průběh z Obr. 6.1 získáme hodnotu *difference* = 160. Při aplikaci na „falešný“ průběh z Obr. 6.1 získáme hodnotu 300. Je tedy patrné, že algoritmus výrazně snižuje hodnotu *difference* s pravděpodobností, že se nejedná o „falešný“ přechod. Obecný postup výpočtu je definován jako:

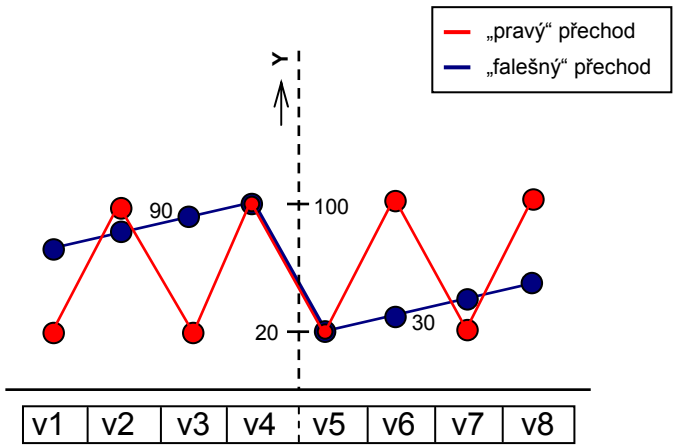
$$d1 = abs(v4 - v5) \cdot 4, \tag{6-1}$$

$$d2 = abs(v3 - v4), \tag{6-2}$$

$$d3 = abs(v5 - v6), \tag{6-3}$$

$$difference = \frac{d1 - d2 - d3}{6}, \tag{6-4}$$

kde *d1* je vážená dynamika hraničních bodů, *d2* a *d3* jsou dynamiky mezi hraničními a prvními mimohraničními body.



Obr. 6.1: Výpočet *difference.*

Abychom ještě více omezili sílu filtrace užitečných přechodů, budeme vypočítanou hodnoutu *difference* limitovat funkcí *GateKeeper*. Její smysl spočívá v omezení *difference* v závislosti na průměrné dynamice „falešných“ přechodů při dané kvalitě. Vstupem funkce *GateKeeper* je *difference* a *Ad* (Average dynamic), výstupem je *Lim*. Funkce je definována jako:

$$Lim = \begin{cases} 0, & difference \le 0 \\ difference, & difference \le Ad \\ 2 \cdot Ad - difference, & difference > Ad \end{cases} \tag{6-5}$$

kde *Lim* je výstupem omezení *difference*, *Ad* je průměrná dynamika „falešných“ přechodů při dané kvalitě komprese.

Výsledná váha, potažmo síla filtrace aktuálního přechodu je definována jako:

$$vaha = abs\left(\frac{Lim}{Ad}\right). \tag{6-6}$$

6.3 Adaptivita – mód hladkých oblastí

Za další stupeň adaptivity filtru můžeme považovat lokální analýzu obrazu, která nám umožní přepínat mezi určitými módy filtru. Tyto módy mohou představovat odlišné druhy filtrace, které dokáží lépe reagovat na detekované lokální vlastnosti obrazu.

Velmi efektivním příkladem je vyhlazování tzv. „hladkých“ oblastí. Ty jsou specifické svou malou dynamikou mezi jednotlivými pixely zkoumaného vektoru. Artefakty vzniklé v těchto oblastech jsou tak vzhledem k malým rozdílům v jejich okolí velmi výrazné. Jak je vidět na Obr. 6.2 a), metody filtrace přechodů využívaných v oblastech s výraznou dynamikou, nemají v „hladkých“ oblastech dostatečnou sílu (účinek). Blokové artefakty tak zůstávají, především ze subjektivního hlediska velmi vizuálně rušivé.

K vyřešení tohoto problému byl vytvořen mód filtru, který na blokové přechody aplikuje silnou filtraci založenou na linearizaci přechodu. K rozhodnutí zdali se jedná o „hladkou“ oblast je nejdříve nutné změřit dynamiku zkoumaného vektoru. To je možné provést vyhledáním pixelu s maximální a minimální úrovní v daném vektoru. Absolutní rozdíl těchto hodnot je hledanou dynamikou, která je dále porovnávána s *Ad* (průměrnou hodnotu „falešných“ přechodů při dané kompresní kvalitě). Pokud je dynamika menší než *Ad* je rozhodnuto o aplikování módu *hladkých oblastí*.

$$PixelMin = \min(v1, v2, v3, v4, v5, v6, v7, v8), \tag{6-7}$$

$$PixelMax = \max(v1, v2, v3, v4, v5, v6, v7, v8), \tag{6-8}$$

$$Mód\_HO = \begin{cases} 1, & (PixelMax - PixelMin) < 0{,}7 \cdot Ad \\ 0, & \text{jinak,} \end{cases} \tag{6-9}$$

kde *v1* až *v8* jsou pixely zkoumaného vektoru, *PixelMin* / *PixelMax* je hodnota pixelu s nejmenší / největší úrovní v testovaném vektoru, *Mód_HO* udává stav módu *hladkých oblastí*, pokud je roven *1* je aktivní, pokud *0* je neaktivní.

Minimalizace blokových artefaktů je v módu *hladkých oblastí* prováděna metodou linearizace přechodu popsanou v 5.7. Pro větší účinnost filtrace je k vypočítané váze přičten offset o velikosti 55%. Váha, která díky přičtenému offsetu překročí hodnotu 1, je limitována. Dalším krokem je již samotný výpočet nových linearizovaných pixelů zkoumaného vektoru. V posledním kroku se tyto nové hodnoty podle vypočítané váhy průměrují s původními. Jak je patrné z Obr. 6.2 b), aplikace módu *hladkých oblastí* velmi výrazně zlepšuje subjektivní vjem vylepšovaného obrazu.

$$vaha = vaha + 0{,}55\,, \tag{6-10}$$

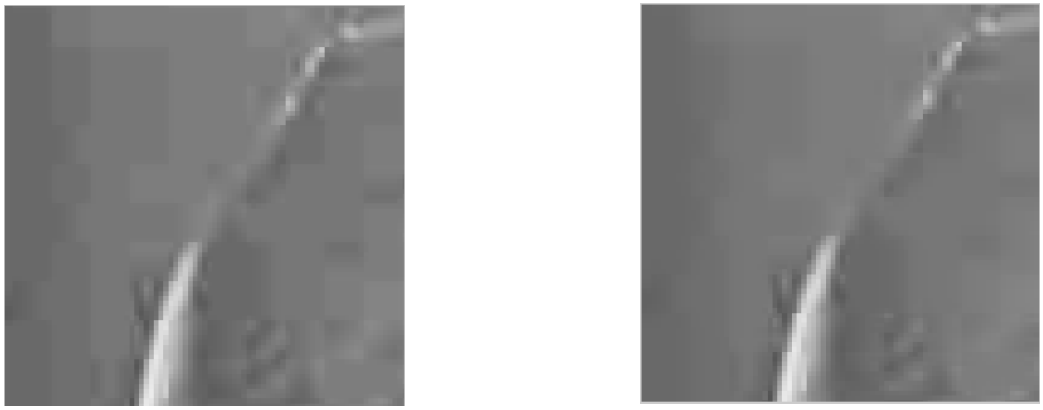
$$vaha = \begin{cases} 1, & vaha > 1 \\ vaha, & \text{jinak,} \end{cases} \tag{6-11}$$

$$d = \frac{(v8 - v1)}{7}, \tag{6-12}$$

$$v_i' = v_1 + (i-1) \cdot d \qquad 1 \le i \le 8, \tag{6-13}$$

$$out_i = v_i' \cdot vaha + v_i \cdot (1 - vaha), \tag{6-14}$$

kde *d* je lineární změna mezi přechody pixelů *v1* až *v8*, v_i' je nový linearizovaný pixel, out_i je výstupní pixel a i je pořadové číslo pixelu ve zkoumaném vektoru [10].



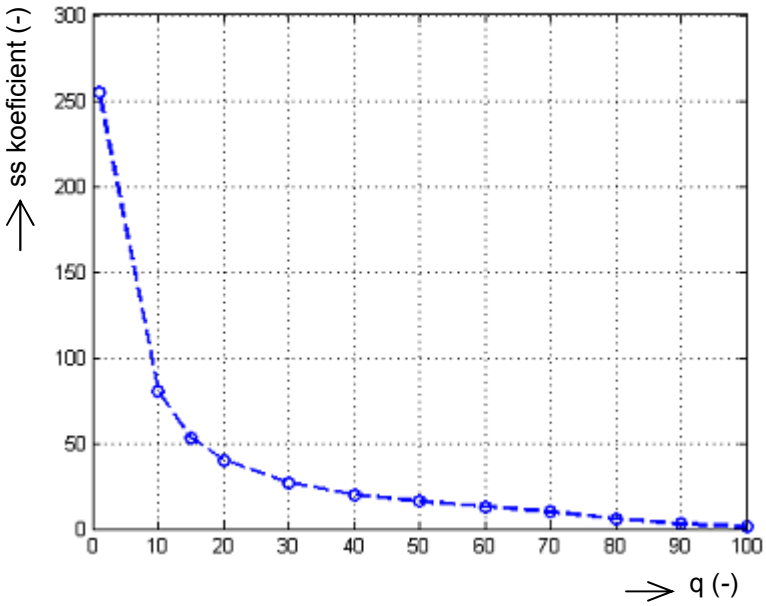
a) b)

Obr. 6.2: Výstup MHP-MBA filtru s módem *hladkých oblastí*,
a) neaktivním, b) aktivním.

6.4 Adaptivita řízená kvantovací tabulkou

Tato adaptivita je velmi důležitá z hlediska získání informace reprezentující kvalitu komprese. Spolu s daty je v soboru typu JPG přenášena hlavička obsahující mimo jiné i kvantovací tabulku, podle které byl daný obraz komprimován. Z prvního koeficientu této tabulky (stejnosměrné složky), tak můžeme zjistit přibližnou kvalitu

(*q*) se kterou byl původní obraz komprimován. Na Obr. 6.3 je zobrazen změřený průběh této závislosti. Takto získaná kvalita (*q*) je odhad pouze z jednoho koeficientu, přesnější hodnotu získáme algebraickým průměrem celé kvantovací tabulky.



Obr. 6.3: Závislost koeficientu kvantovací tabulky na kvalitě komprese „*q*“.

6.5 Minimalizace přechodu

V 6.2 jsme popsali postup jakým je v MHP-MBA filtru detekován „falešný“ přechod spolu s výpočtem váhy, která určuje sílu jeho následné minimalizace. Způsobem jakým tato minimalizace probíhá, se budeme zabývat právě nyní.

Použitá metoda minimalizace vychází z myšlenky nastavit dva hraniční body přechodu tak, aby byl jejich rozdíl roven průměrnému rozdílu mezi jednotlivými prvky levé, či pravé strany. Proto vektor *v1* až *v8* rozdělíme na levou a pravou část a to symetricky vůči přechodu mezi bloky. V dalším kroku vypočítáme váženou[2] průměrnou změnu pixelů v pravé a levé straně, čímž získáme *Ad_L*, *Ad_R* (Average different – Left / Right). Z těchto hodnot vybereme tu menší, čímž zabráníme příliš velké změně mezi pixely nově vypočítaného přechodu. Tímto postupem jsme tedy získali hodnotu *Ad_min* vyjadřující menší z průměrných změn levé a pravé strany testovaného vektoru. Tato hodnota se stane rozdílem mezi pixely nového přechodu.

[2] Hodnoty změn pixelů mají tím větší váhu, čím jsou blíže přechodu.

Nejdříve je ovšem potřeba vypočítat průměrnou hodnotu prvních mimohraničních bodů *d* od které se následně odečte/přičte polovina *Ad_min*, čímž nám vzniknou nové body přechodu $v5', v4'$. Původní a minimalizovaný průběh vektoru přes přechod je zobrazen na Obr. 6.4. Popsaný postup můžeme definovat jako:

$$Ad\_L = fix\left(\frac{abs(v1-v2)+2\cdot abs(v2-v3)+3\cdot abs(v3-v4)}{6}\right), \tag{6-15}$$

$$Ad\_R = fix\left(\frac{3\cdot abs(v5-v6)+2\cdot abs(v6-v7)+abs(v7-v8)}{6}\right), \tag{6-16}$$

$$Ad\_\min = \min(Ad\_R, Ad\_L), \tag{6-17}$$

$$Ad\_\min = \begin{cases} -Ad\_\min, & v4 < v5 \\ Ad\_\min, & v4 \geq v5 \end{cases} \tag{6-18}$$

$$d = \frac{v4+v5}{2}, \tag{6-19}$$

$$v4' = d + fix\left(\frac{1}{2}\cdot Ad\_\min\right), \tag{6-20}$$

$$v5' = d - fix\left(\frac{1}{2}\cdot Ad\_\min\right), \tag{6-21}$$

$$v6' = v6 - \left(\frac{1}{10}\cdot Ad\_\min\right), \tag{6-22}$$

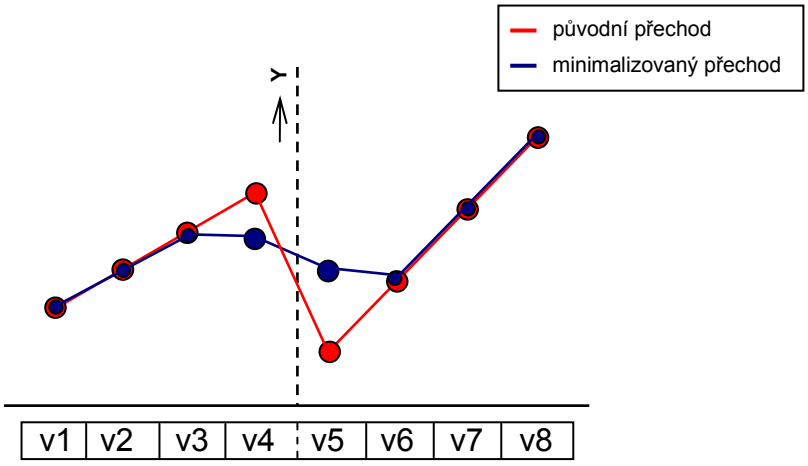
$$v3' = v3 + \left(\frac{1}{10}\cdot Ad\_\min\right), \tag{6-23}$$

kde *Ad_L* / *Ad_R* je průměrná vážená změna pixelů v levé / pravé straně testovaného vektoru, *Ad_min* je menší z hodnot *Ad_L*, *Ad_R*, *d* je průměrná hodnota mimohraničních pixelů *v3*, *v6* a $v3', v4', v5', v6'$ jsou nové hodnoty hraničních a prvních mimohraničních pixelů.

Ve výše popsaném postupu jsme nepoužili vypočítanou váhu určující sílu filtrace. Tu využijeme teprve v posledním kroku, kde podle ní průměrujeme nově vypočítané pixely přechodu se starými a to podle:

$$\mathrm{out_i} = \mathrm{round}\left(\frac{2 \cdot v_i' \cdot vaha + v_i + (1 - vaha)}{vaha + 1}\right) \qquad 3 \le i \le 6, \tag{6-24}$$

kde out_i je nový výstupní pixel a *i* je pořadové číslo pixelu v testovaném vektoru obrazu.



Obr. 6.4: Minimalizace přechodu v MHP-MBA filtru.

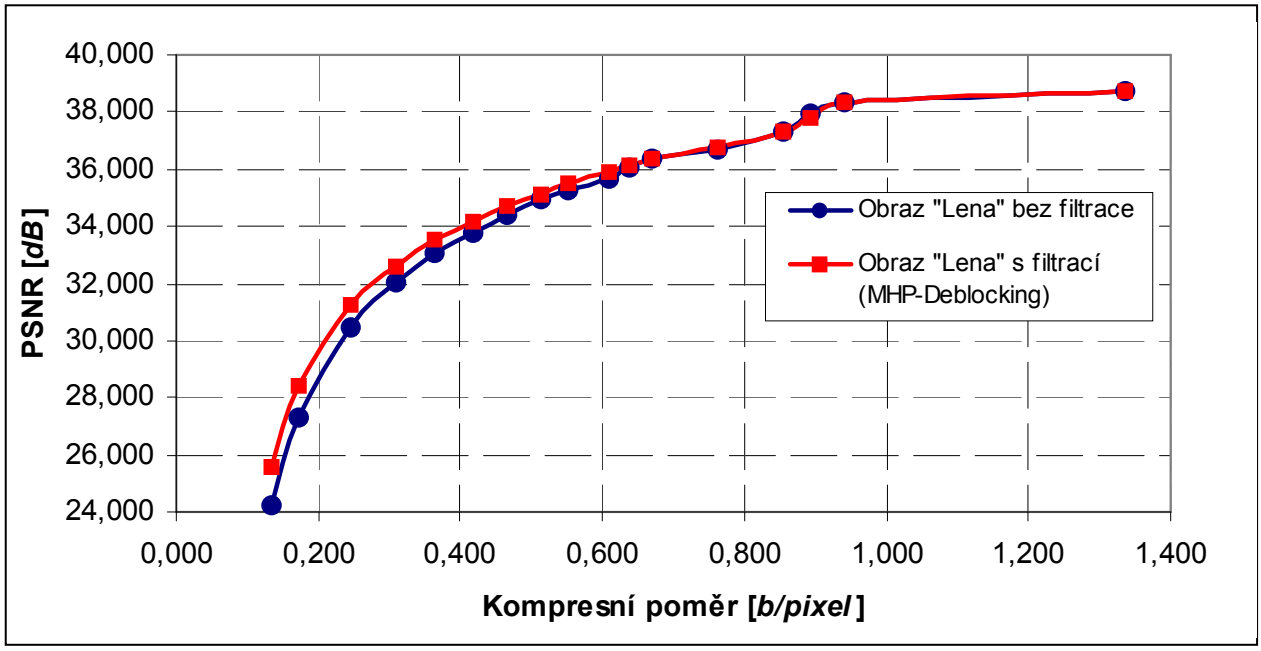
Dalšího zvýšení kvality filtrace jsme dosáhli porovnáváním, zdali se nově vypočítané pixely přechodu příliš neliší od původních. Pokud je rozhodnuto o tom, že se nový pixel liší příliš, aplikuje se na ten původní filtrace dolní propustí s maskou $\left[\frac{1}{5} \quad \frac{3}{5} \quad \frac{1}{5}\right]$. Rozdíl nového a původního pixelu se porovnává s hodnotou *Lim*, která je násobena konstantou 0,6. Tato konstanta je výsledkem objektivního měření (PSNR) a empirického testování.

$$out_i' = \begin{cases} \dfrac{v_{i-1} + 3 \cdot v_i + v_{i+1}}{5} & abs(out_i - v_i) > (0{,}6 \cdot Lim) \\ out_i & \text{jinak,} \end{cases} \tag{6-25}$$

kde out_i' je výstupní hodnota nového pixelu, *Lim* je výstupem po omezení *difference* a *i* je pořadové číslo pixelu přechodu (*4* nebo *5*).

6.6 Výsledky filtru MHP-MBA

Navrhnutý MHP-MBA filtr byl testován na obrazu „Lena.jpg“. Ten byl komprimován kvalitou v rozmezí 1 až 100 *q*, což představuje hodnoty koeficientů kvantovací tabulky od 255 do 1. Pro porovnání kvality výstupních pramenů byl zvolen deblocking filtr z video kodeku H.263. Jako referenční filtr nebyl zvolen náhodně, ale díky svým parametrům, které jsou podobné našim vstupním požadavkům. Mezi ně patří především co možná nejmenší výkonová náročnost. Ta je u H.263 nezbytná vzhledem k využití ve video sekvencích.



Obr. 6.5: Závislost kvality obrazu na kompresním poměru před a po filtraci.

V Tab. 6-1 jsou uvedeny naměřené hodnoty PSNR pro 18 obrazů „Lena.jpg“, komprimovaných s kvalitou 1 až 85. Každý z těchto obrazů byl vylepšován jak MHP-MBA filtrem, tak deblocking filtrem z H.263. Jak je z Tab. 6-1 patrné má MHP- MBA filtr výrazně lepší objektivní vlastnosti než H.263 především v pásmu kompresní kvality od q = 10, až 40. Především v tomto pásmu je obecně deblocking filtr nejvíce vizuálně užitečný. Při ještě nižší kvalitě (q < 10) je daný obraz již příliš znehodnocen a tím i velice těžce použitelný. Naopak v pásmu q > 40 je výskyt blokových artefaktů

velmi málo patrný a deblocking filtr tak ztrácejí svůj význam. V pásmu $q > 70$ dokonce využití těchto filtrů způsobuje degradaci objektivní kvality. Z tohoto důvodu je do MHP-MBA filtru zaveden ochranný prvek, který při kvalitě nad 70 nastaví hodnotu *Ad* na 0.1, čímž se prakticky deaktivují jeho filtrační schopnosti.

Tab. 6-1: Výsledky měření objektivní kvality filtru MHP-MBA.

Kvalita komprese q [-]	**1**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
Kompresní poměr [b/pixel]	**0,134**	**0,173**	**0,246**	**0,309**	**0,364**	**0,417**	**0,467**	**0,513**	**0,552**
PSNR [dB] (*bez filtrace*)	24,256	27,345	30,445	32,003	33,042	33,806	34,410	34,926	35,277
PSNR [dB] (*MHP-Deblock*)	25,564	**28,394**	**31,253**	**32,625**	**33,505**	**34,186**	**34,686**	**35,148**	**35,477**
PSNR [dB] (*H.263*)	25,570	28,331	31,120	32,483	33,395	34,096	34,642	35,105	35,422
Kvalita komprese q [-]	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**
Kompresní poměr [b/pixel]	**0,611**	**0,639**	**0,668**	**0,762**	**0,856**	**0,892**	**0,941**	**1,337**	**1,624**
PSNR [dB] (*bez filtrace*)	35,702	36,072	36,357	36,670	37,300	37,929	38,341	38,746	43.6
PSNR [dB] (*MHP-Deblock*)	**35,865**	**36,176**	36,401	36,743	37,311	37,808	**38,341**	**38,746**	**43.6**
PSNR [dB] (*H.263*)	35,849	36,150	36,419	36,755	37,357	37,864	38,160	38,657	43.192

Ze subjektivního hlediska má obraz po průchodu MHP-MBA filtrem výrazně vyšší kvalitu, což je dobře patrné na Obr. 6.6. Pokud budeme porovnávat visuální kvalitu obrazu vylepšeného MHP-MBA filtrem se stejným obrazem vylepšeným filtrem z H.263 zjistíme, že kvalitativně lepší je výstup MHP-MBA filtru. To je způsobeno především díky detekci „hladkých oblastí“ a jejich následné silné filtraci.

Jak již bylo popsáno výše, důležitým parametrem realizovaného filtru je jeho nízká výkonová náročnost. Tu se jako nejefektivnější ukázalo měřit ve srovnání opět s deblocking filtrem z H.263. V Tab. 6-2 jsou výsledky 17 měření doby, která byla potřeba pro vylepšení obrazu daným filtrem. Z výsledku aritmetického průměru těchto hodnot vyplívá, že náročnost MHP-MBA filtru je přibližně o 40%[3] vyšší než je tomu u H.263. Vzhledem k nárůstu kvality výstupu a celkové větší složitosti realizovaného filtru, je tento nárůst relativně malý.

[3] Platí pro platformu PC (MATLAB).

Tab. 6-2: Měření výkonové náročnosti MHP-MBA filtru oproti H.263.

měření č.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
H.263 [s]	1,13	1,06	1,05	1,06	1,06	1,07	1,07	1,06	1,05
MHP-Deblocking [s]	1,61	1,50	1,47	1,48	1,48	1,49	1,46	1,47	1,47
Δ [%]	42,63	41,95	39,75	40,40	39,68	39,56	35,77	38,72	39,27
měření č.	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	
H.263 [s]	1,06	1,07	1,06	1,06	1,06	1,07	1,07	1,07	
MHP-Deblocking [s]	1,47	1,46	1,45	1,46	1,46	1,46	1,44	1,58	
Δ [%]	39,16	37,18	37,22	37,74	37,63	36,70	34,87	47,86	
$(\sum\Delta)/17$ [%]	**39,18**								

a) b)

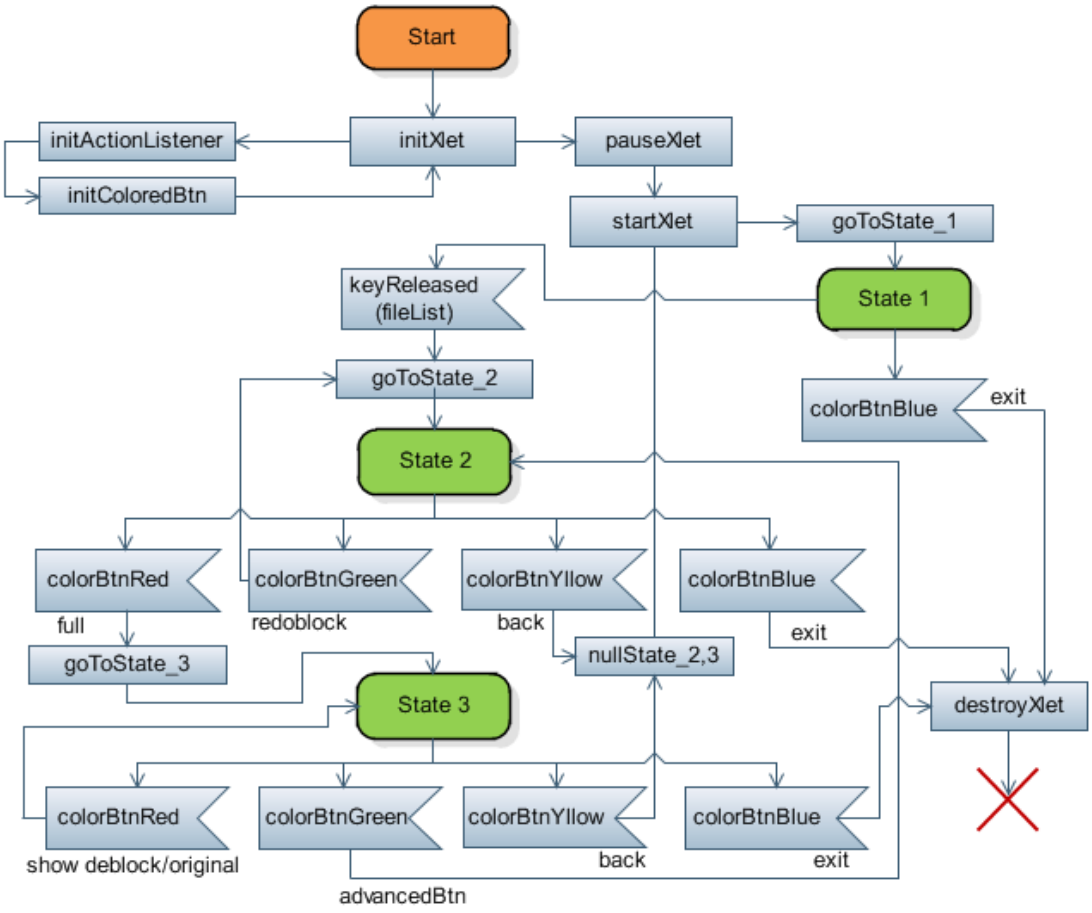
Obr. 6.6: Obrázek „Lena.jpg“, a) bez filtrace, b) s použitím MHP-MBA filtru.

7 Aplikace MHP-Deblocking

7.1 Struktura aplikace MHP-Deblocking

Prvním krokem při vývoji aplikace MHP-Deblocking, bylo vytvoření filtru MHP-MBA s požadovanými vlastnostmi a to v prostředí MATLAB. Druhým krokem bylo vytvoření samotné Xlet aplikace MHP-Deblocking, která v sobě integruje právě filtr MHP-MBA.

Strukturu aplikace MHP-Deblocking můžeme rozdělit do tří základních stavů. Přechod do těchto stavů zajišťují funkce: `goToState_1`, `goToState_2`, `goToState_3`. Jak je vysvětleno v kapitole 3.6, mají Xlet aplikace podobnou strukturu jako webové aplety. Aplikace je proto nejdříve inicializována prostřednictvím metody `initXlet`, po jejímž dokončení přejde do stavu `pauseXlet`. Z tohoto stavu již může být aplikace pomocí metody `startXlet` spuštěna.



Obr. 7.1: Struktura aplikace MHP – Deblocking.

Úkolem metody `initXlet` je především nastavení komponenty `HScene`, která určuje nejen vlastnosti (rozměry) zobrazované plochy, ale spravuje všechny další grafické komponenty. Prostřednictvím této komponenty je také řízen životní cyklus celého programu. Další úlohou metody `initXlet` je inicializace událostí na stisk tlačítek a nastavení navigačních tlačítek. Aplikace je spuštěna zavoláním metody `startXlet`, která zobrazí komponentu `HScene` a dále zavolá funkci `goToState_1` po jejímž dokončení je aplikace ve stavu 1.

Grafické rozhraní tohoto stavu je zobrazeno na Obr. 7.3. Základním prvkem je grafická komponenta `fileList`, prostřednictvím které je možné vybrat jednotlivé JPG soubory. Dalšími prvky je zobrazení náhledu souboru, na němž je v daný okamžik kurzor a funkční tlačítka reprezentující 4 barevná tlačítka na dálkovém ovládání set-top-boxu. Zobrazená tlačítka mají informativní charakter a uživateli pouze oznamují aktuální činnost. Vlastní funkce jsou spuštěny v závislosti na vyvolání klávesových událostí nad grafickými komponentami. Ve stavu 1 je aktivní pouze modré funkční tlačítko, jehož vyvoláním je spuštěna metoda `destroyXlet`, která celou aplikaci ukončí. Při zvolení požadovaného souboru je vyvolána klávesová událost `keyReleased`, která spustí funkci `goToState_2`, po jejímž dokončení vstoupí aplikace do stavu 2.

V tomto režimu je aplikace určena pro nastavení filtrace a kontrolu, porovnání jejich výsledků. Pomocí radiobuttonů `MBA`, `H.263` se nastavuje metoda, která bude použita pro deblocking. Prostřednictvím checkBoxu `PSNR / MSE` je zapínán/vypínán následný výpočet PSNR (Peak Signal-to-Noise Ratio) a MSE (Mean Squared Error) vylepšeného obrazu v porovnání s originálním nekomprimovaným obrazem[4]. V komponentě `HSinglelineEntry` je defaultně zobrazena hodnota kvality komprese vylepšovaného obrazu `Q`, která byla detekována ve zdrojovém souboru. Tuto hodnotu může uživatel, v rozsahu 1 až 100, libovolně měnit (čímž dojde k přenastavení prahových hodnot v celém filtru) a následně sledovat jaký vliv má na výsledky filtrace. Rozhraní dále zobrazuje 3 náhledy:

- zdrojový nevylepšený obraz,
- výsledný obraz s minimalizovanými blokovými artefakty,
- obraz, v němž jsou zvýrazněny oblasti s detekovanou malou dynamikou[5].

[4] Výpočet proběhne pouze, pokud je originální nekomprimovaný obraz k dispozici.
[5] Zobrazuje se pouze při použití filtru MHP-MBA.

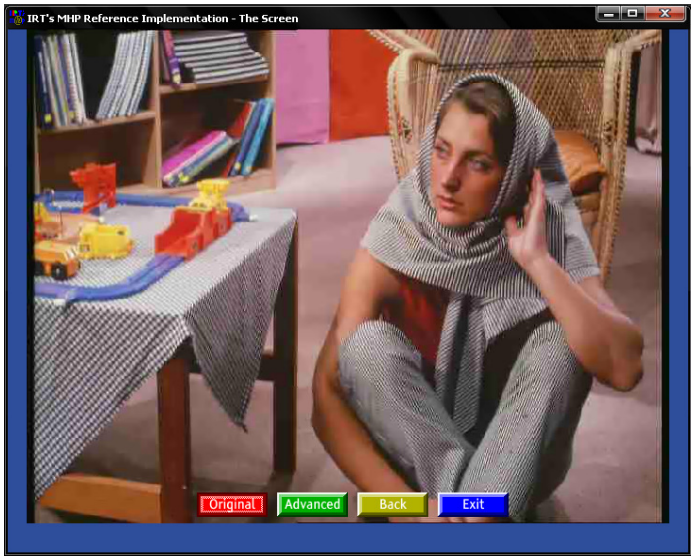
Protože jsou tyto náhledy většinou příliš malé pro srovnání detailních změn, byly do grafického rozhraní implementovány kurzory umožňující posun zobrazovaného výřezu a dále zoomovací kurzory, které umožňují libovolnou změnu zvětšení tohoto výřezu. Posledním grafickým prvkem jsou barevná tlačítka, která mají v tomto stavu funkci:

- **červené** (Full) – zobrazí vylepšený obraz na celé obrazovce, přechod do stavu 3,
- **zelené** (ReDeblock) – opětovně spustí filtraci vybraného obrazu s aktuálním nastavením,
- **žluté** (Back) – umožní výběr nového obrazu, přechod do stavu 1,
- **modré** (Exit) – ukončí aplikaci.

Po přechodu aplikace do stavu 3 má být přes celou plochu obrazovky zobrazen vylepšený obraz. Jak je ovšem vidět z Obr. 7.2, který byl pořízen ze simulátoru IRT není obraz roztažen úplně do krajů. To je zcela záměrné a vyhneme se tím nechtěnému oříznutí, které způsobuje zobrazení na skutečné televizi. Jedinými dalšími zobrazenými grafickými komponentami v stavu jsou opět barevná tlačítka tentokrát ve funkci:

- **červené** (Originál/Deblock) – přepíná mezi zobrazením původního a vylepšeného obrazu,
- **zelené** (Advanced) – přepne aplikaci do pokročilého módu, tedy do stavu 2.

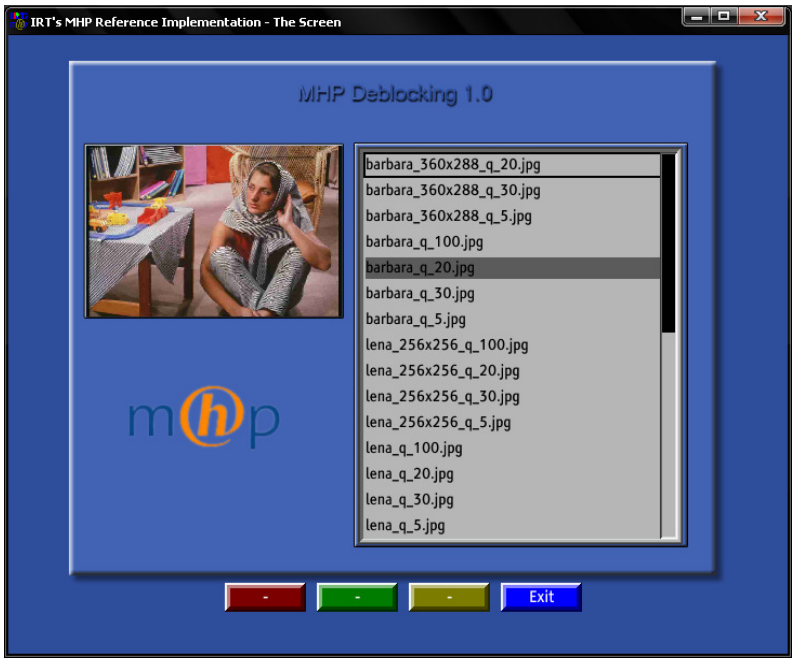
Další dvě tlačítka mají stejnou funkci jako ve stavu 2.



Obr. 7.2: Aplikace MHP-Deblocking ve stavu 3.

7.2 Stav 1 – funkce goToState_1

Do stavu 1 přejde aplikace MHP-Deblocking po úspěšném dokončení metody `goToState_1`. Ta má za úkol inicializovat a nastavit grafickou komponentu `HListGroup`, dále načíst a zobrazit souboru s příponou JPG, které byly, spolu s aplikací, nahrány prostřednictvím objektového karuselu DSM-CC. Názvy těchto souboru jsou přidány jako jednotlivé položky do objektu `HListGroup`. Tento objekt dále umožní uživateli procházet položkami a zvolit soubor, který má být zpracován. Aby bylo možné reagovat na uživatelovo vybrání souboru, je nutné nejprve aplikovat naslouchače klávesových událostí, což je provedeno pomocí funkce `initItemListner()`. Poslední prováděnou funkcí je `setColoredBtnToState_1()`, která nastaví navigační tlačítka do stavu 1. Navigační tlačítka reprezentují čtyři barevné klávesy (červená, zelená, žlutá, modrá) na dálkovém ovládání SetTopBoxu. V aplikaci jsou využity k ovládání nejdůležitějších funkcí celého programu a jejich jednotlivé funkce se mění v závislosti na stavu, ve kterém se aplikace nachází. O jejich momentální funkci je uživatel informován pomocí popisek čtyř grafických komponent `HTextButton` zobrazených ve stejných barvách. Samotná funkce daného navigačního tlačítka je vyvolána prostřednictvím klávesových událostí reagujících na identifikační kódy stisknutých „barevných“ tlačítek. Funkce `goToState_1` je ukončena po nastavení proměnné `state` na hodnotu 1. Tato proměnná je určující při zjišťování v jakém stavu se aplikace v dané chvíli nachází.



Obr. 7.3: Aplikace MHP-Deblocking ve stavu 1.

7.3 Stav 2 – funkce goToState_2

Stěžejní funkcí celého programu je `goToState_2`, která provádí samotnou minimalizaci blokových artefaktů. Její spuštění vyvolá uživatelovo vybrání položky komponenty `HListGroup`, představující určitý JPG soubor. Adresa tohoto souboru je uložená a následně využita k jeho načtení.

Funkce je volána s parametrem `isInitGUI`, jehož hodnota rozhoduje, zda-li bude spuštěna funkce `initGUIstate2`. Ta zajišťuje inicializaci grafických komponent a jejich nastavení. Toto nastavení je potřeba provést pouze jednou a proto se při dalším pokusu o přechod do stavu 2 již `initGUIstate2` nespouští.

Dále je načten a zobrazen originální (nevylepšený) obraz pomocí objektu třídy `ShowPanel_resize`, který je naplněn adresou vybraného souboru a parametry umožňující následnou změnu velikosti a pozice výřezu zobrazovaného obrazu. Samotné načtení obrazových dat z paměti set-top-boxu zajišťuje objekt třídy `MediaTracker`, který vrátí objekt třídy `Image`. Z hlavičky obrazového souboru je prostřednictvím objektu třídy `ImageQuality` analyzována kvalita komprese `Q`. Pro získání elementárních obrazových dat je následně prostřednictvím objektu třídy `ImageToMatrix` převeden načtený obraz do pole, které je v dalším kroku přepočítáno z barevného prostoru RGB do prostoru YUV.

Obr. 7.4: Aplikace MHP-Deblocking ve stavu 2.

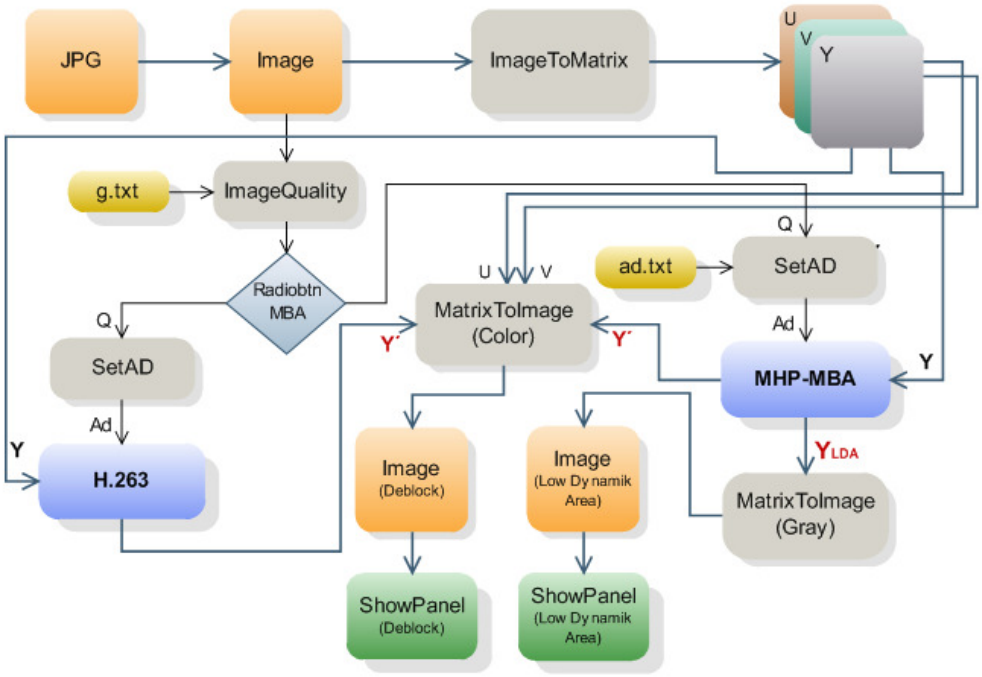
V aplikaci je implementována, mimo vytvořené deblocking metody MHP-MBA, i metoda převzatá z kodeku H.263. O tom, která z těchto metod bude použita, rozhoduje podmínka testující, zda-li je stisknuto tlačítko `radiobtn_MBA` třídy `HToggleButton`. Je-li stisknuto, spustí se funkce `getAdMBA` třídy `SetAD`, která podle parametru `Q` vyhledá v souboru „ad.txt“ příslušnou hodnotu `Ad` (Average different = průměrná intenzita blokových artefaktů při dané kvalitě). Tato hodnota je s polem jasové složky `Y` předána metodě `deblocking_MBA` třídy `Deblock_MHP_MBA`, která provede samotnou minimalizaci blokových artefaktů. Její návratový objekt je opět pole jasových složek (`Y´`), ovšem již s filtrovanými artefakty, které je ukládáno opět do vstupního pole, čímž se šetří paměťové nároky procesu filtrace.

Upravené jasové složky jsou v dalším kroku předány, spolu s původními barevnými složkami `U`, `V`, metodě `createImageColorYUV` třídy `MatrixToImage`, která z nich zpětně vytvoří objekt typu `Image`. Tento objekt již může být zobrazen na obrazovce v kontejneru `showPanel_deblock`.

Druhým návratovým objektem metody `deblocking_MBA` je pole jasových složek obsahující hodnotu 255 (plný jas) na pozicích ve kterých filtr analyzoval oblasti s nízkou dynamikou. Toto pole je opět předáno objektu třídy `MatrixToImage`, tentokrát ovšem metodě `createImageGrey`, která z něj vytvoří černobílí objekt `Image`. Ten je následně zobrazen v kontejneru `showPanel_lowDynamicArea`.

Pokud výše popsaná podmínka vyhodnotí, že není tlačítko `radiobtn_MBA` stisknuté, spustí se deblocking metoda z kodeku H.263. Prvním krokem je opět získání parametru `Ad` prostřednictvím třídy `SetAd`, nyní ovšem s použitím funkce `getAdH263`, která `Ad` vypočítá použitím převodní funkce. Tento parametr je spolu jasovou složkou `Y` předán metodě `deblocking_H263` třídy `Deblock_H263`. Tato metoda provede minimalizaci blokových artefaktů a následně vrátí pole upravených jasových složek `Y´`.

Toto pole je stejně jako v metodě MBA předáno, spolu s původními barevnými složkami `U`, `V`, funkci `createImageColorYUV`, která z něj vytvoří objekt `Image` a následně zobrazí v kontejneru `showPanel_deblock`.



Obr. 7.5: Průběh zpracování a filtrování obrazu.

7.4 *Analyzování kvality komprese – třída ImageQuality*

Aby bylo možné řídit správně průběh filtrace, je nutné vědět jakou kvalitu komprese má vylepšovaný obraz. Zjištění této informace má na starosti objekt třídy `ImageQulity`, respektive jeho návratová funkce `getQ`. Způsob analýzy je založen na znalosti struktury obrazového kontejneru JFIF (JPG). Ten je složen jak z obrazových dat, tak z hlavičky ve které je uložena, krom jiných informací, kvantovací tabulka, s níž byl komprimován originální obraz [15].

Prvním krokem analýzy je zjištění průměrné hodnoty koeficientů kvantovací tabulky. To zajišťuje metoda `getQuant`, která po svém zavolání začne číst obrazový soubor do té doby, dokud nenalezne značku `0xff`, další bajt nás informuje o tom, jaká data budou v hlavičce souboru následovat. Metoda proto pokračuje hledáním příznaku `0xDB`, který nás informuje o začátku kvantovací tabulky. Další dva bajty nám oznamují kolik má tato tabulka koeficientů. Následuje cyklus, ve kterém jsou tyto koeficienty vyčítány a jejich algebraický součet je vrácen metodě `getQ` [15].

Druhým korkem analýzy je samotné přiřazení hodnoty kvality `Q`. K řešení tohoto úkolu byl využit kodér „jpg“ obrazů implementovaný v programu MATLAB. Pomocí něho bylo vytvořeno 100 obrazů komprimovaných s kvalitou v rozmezí 1 až 100. Pro každý z těchto obrazů byla pomocí metody `getQuant` zjištěna průměrná hodnota

koeficientů kvantovací tabulky, k nimž byla přiřazena příslušná kvalita `Q`, se kterou byl daný obraz komprimován. Takto získané informace byly uloženy do souboru „q.txt“. Metoda `getQ` následně z těchto informací hledá tu nejpodobnější hodnotu průměru koeficientů kvantovací tabulky k hodnotě získané z vylepšovaného obrazu. Pořadové číslo řádku v souboru „q.txt“ na kterém se cyklus zastaví je námi hledaná hodnota `Q`.

Ačkoliv jsou referenční hodnoty v souboru „q.txt“ získány pouze z „jpg“ kódéru implementovaného v programu MATLAB, odpovídá odhad hodnoty `Q` i v případě ve kterém byl soubor komprimován pomocí jiného programu (např. Fotoshop).

7.5 Výpočet parametru Average different – třída SetAd

Hodnotu `Q`, kterou jsme získali z vylepšovaného obrazu, nemůžeme použít přímo pro řízení deblocking filtru. Intenzita blokových artefaktů není totiž v rozmezí kvality 1 až 100 lineární. Nejdříve je proto nutné vypočítat hodnotu `Ad` (Average different), která představuje průměrnou hodnotu přechodu blokových artefaktů při dané kvalitě `Q`. K tomuto úkolu byla vytvořena třída `SetAd` obsahující metodu `getAdMBA`. Pro samotný výpočet hodnoty `Ad` byly použity dva odlišné způsoby.

Tím prvním byl převod na základě převodní funkce. Tato funkce je odvozena z testovacích obrazů, které byly filtrovány s různou hodnotou `Ad`. Z takto získaných výsledných obrazů byly vybrány ty s nejlepší kvalitou (největší PSNR) vůči originálnímu nekomprimovanému obrazu, čímž mohla být z příslušných hodnot `Ad` aproximována výsledná převodní funkce. Výhodou tohoto způsobu je jednoduchost. Nevýhodou je nepřesnost výpočtu „nejlepších“ hodnot `Ad` v celém průběhu převodní funkce, což snižovalo účinnost filtrace při určitých kvalitách `Q`. Z tohoto důvodu byl vytvořen druhý způsob převodu.

Ten je založen stejně jako v metodě `getQ` na testování sta obrazů s kvalitou v rozmezí 1 až 100. Každý z těchto obrazů byl opět filtrován s různou hodnotou `Ad`, z nichž byl vybrán ten s nejlepším výsledkem měření PSNR. Tímto postupem jsme získali 100 hodnot `Ad`, které byly s příslušnými hodnotami `Q` uloženy do souboru „ad.txt“. Výsledná metoda `getAdMBA(Q)` třídy `SetAd` tedy pouze prohledává soubor „ad.txt“ do té doby dokud nenalezne řádek obsahující značku kvality `Q` vylepšovaného obrazu, na tomto řádku je uložena i hledaná hodnota `Ad`.

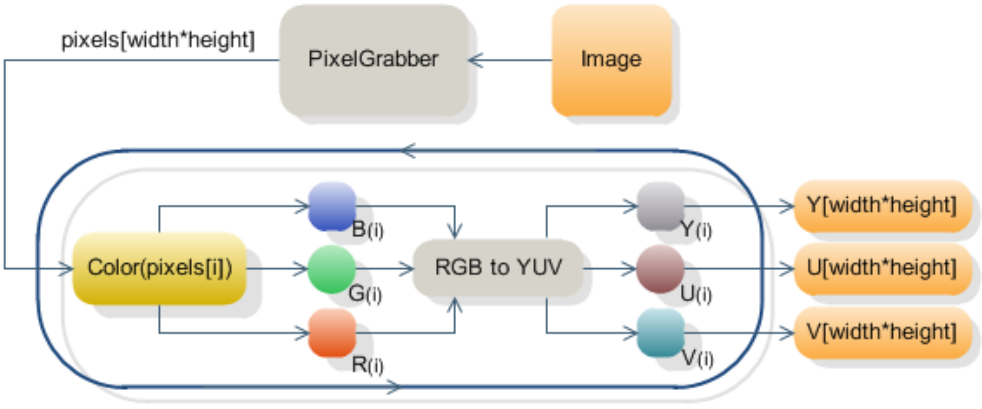
7.6 Výpočet reprezentace obrazu – třída ImageToMatrix

Aby bylo možné jakýmkoli způsobem upravovat, či filtrovat obraz uložený v objektu `Image`, je nejdříve nutné získat reprezentaci jeho jednotlivých bodů. To je možné provést dvěma způsoby.

Prvním je použití objektu třídy `DVBBufferedImage`, který již obsahuje metody `getRGB(x,y)`, `setRGB(x,y)` umožňující přístup a změnu jednotlivých pixelů obrazu. Díky tomu že je pro čtení i pro zápis obrazových dat použita pouze jedna instance třídy `DVBBufferedImage`, má tato metoda malou paměťovou náročnost. Naopak nevýhodou je relativně velké zpoždění při inicializaci objektu a přístupu k datům.

Druhým možným způsobem je využít objekt třídy `PixelGrabber`, který pomocí jeho metody `grabPixels` převede objekt `Image` do matice typu `Integer`. Tato matice, přesněji řečeno vektor, má velikost `Integer[width*height]`, kde `width` je počet sloupců a `height` počet řádků převáděného obrazu. Každá hodnota vektoru představuje RGB reprezentaci příslušného pixelu. Důvodem proč není použito dvourozměrné matice, je velká doba zpoždění při její inicializaci.

Jak bylo zmíněno, jsou získané body obrazu v barevném prostoru RGB. Každý bod je tedy složen ze tří základních barev (červená, zelená, modrá) z nichž každá nabývá hodnoty v rozsahu 0 až 255. Pokud bychom obraz filtrovali přímo v tomto barevném prostoru, museli bychom tuto filtraci provést pro každou z jeho barevných složek (tedy 3krát).



Obr. 7.6: Metoda `getMatrixYUV` třídy `ImageToMatrix`.

Z hlediska výkonu je proto výhodnější využít systém YUV. Ten byl vyvinut pro televizní vysílání, ve kterém vznikla potřeba kompatibility mezi černobílým a barevným vysíláním. To bylo možné zajistit oddělením jasové (Y) a barevné složky (U, V), což můžeme s výhodou využít i ve zpracování obrazu, kdy upravujeme (filtrujeme) pouze jasovou část, zatímco složky barevné jsou následně beze změny sloučeny s výslednou (upravenou) jasovou složkou. Oproti zpracování v prostoru RGB, tak ušetříme 2 třetiny času procesoru.

Jak již bylo zmíněno, každý prvek vektoru `pixels` získaného třídou `PixelGrabber` reprezentuje jeden RGB bod obrazu. Při konverzi barevných prostorů musíme nejdříve rozdělit tyto prvky na jednotlivé složky (R,G,B). To je možné provést objektem třídy `Color`, který naplníme jednotlivými prvky vektoru `pixels`. Tento objekt obsahuje metody `getRed`, `getGreen`, `getBlue`, které vrátí hodnotu jednotlivých složek v rozsahu 0 až 255. Dále již následuje samotný převod do prostoru YUV a to dle následujících vzorců:

$$Y_i = fix\left(0{,}299 \cdot R_i + 0{,}587 \cdot G_i + 0{,}114 \cdot B_i\right), \tag{7-1}$$

$$U_i = fix\left[\left(B_i - Y_i\right) \cdot 0{,}492\right], \tag{7-2}$$

$$V_i = fix\left[\left(R_i - Y_i\right) \cdot 0{,}877\right], \tag{7-3}$$

kde Y_i je jasová a U_i, V_i barevné složky pixelu, R_i červená složka pixelu, G_i zelená složka pixelu, B_i modrá složka pixelu a i je pořadové číslo obrazového bodu ve vektoru `pixles`.

Zaokrouhlování *fix* je použito z důvodu převodu výstupních vektorů z typu **`Double`** (64 bitů) na typ **`Short`** (16 bitů), čímž je ušetřena velká část potřebné paměti. Nevýhodou tohoto zaokrouhlení je nemožnost následného přesného zpětného výpočtu RGB hodnot. Takto vytvořená chyba je ovšem nepatrná a lidským zrakem nepostřehnutelná.

Výstupem metody `getMatrixYUV` třídy `ImageToMatrix` jsou vektory **`short`**`[] Y`, **`short`**`[] U` a **`short`**`[] V`.

7.7 Třída Deblock_MHP_MBA

Nyní již máme všechna data potřebná pro spuštění samotné filtrace minimalizující blokové obrazy. Ta je prováděna objektem třídy `Deblock_MHP_MBA`, respektive jeho metodou `deblocking_MBA`. Tato metoda je dvojnásobně přetížená, nejdříve ve funkci pracující s obrazovými daty prostřednictvím objektu třídy `DVBBufferedImage` se vstupními parametry ve tvaru:

```
deblocking_MBA(DVBBufferedImage buffImage, int ad),
```

kde `buffImage` je objekt typu `DVBBufferedImage` obsahující obrazová data vylepšovaného obrazu a „ad“ je detekovaný parametr Average Dynamic. Dále metoda může pracovat ve funkci pracující s jasovou složkou obrazu uloženou v poli typu `short[]`:

```
deblocking_MBA(short[] inMatrix, int ad, int width, int height,
short[] lowDynamicMatrix),
```

kde `inMatrix` je pole typu **`short`**`[]` obsahující jasovou složku vylepšovaného obrazu, `width`/`height` což je počet sloupců/řádků tohoto obrazu a pole `lowDynamicMatrix` do kterého budou zaznamenány ty oblasti obrazu, ve kterých bude detekována malá dynamika.

Jak se ukázalo z pozdějšího testování, je metoda využívající objekt typu `DVBBufferedImage` výrazně výkonově náročnější a proto se ve finální verzi nepoužívá. Dále se proto budeme zabývat pouze metodou využívající pole typu **`short`**`[]`.

Základním principem minimalizace blokových artefaktů ve vytvořeném filtru je analyzování až osmy pixelů kolmo protínajících minimalizovaný přechod. Tuto analýzu, stejně jako následný výpočet nových bodů, provádí filtr nejdříve v horizontálním a následně ve vertikálním směru. Kód pro oba tyto průchody je, až na rozdíl v načítání a ukládání jednotlivých obrazových pixelů, totožný. Pro zajištění větší přehlednosti a zajištění stejné funkce filtru v obou průchodech je jak pro horizontální, tak pro vertikální filtrování volána stejná funkce `core_MBA`, která je zobrazena na Obr. 7.7.

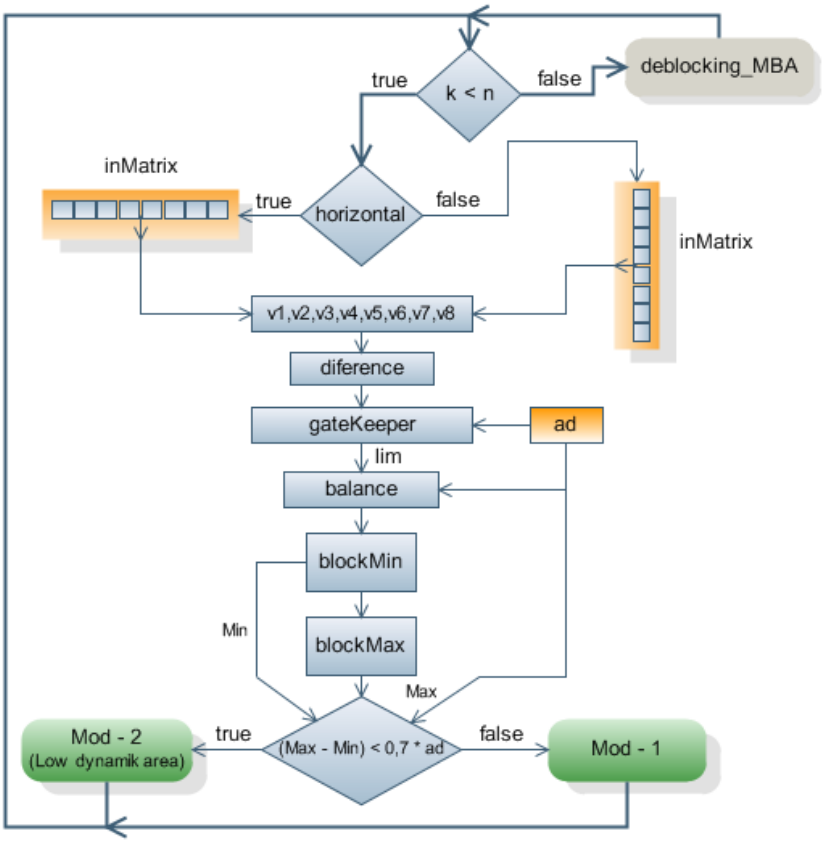
Po vyvolání metodou `deblocking_MBA` vstoupí funkce `core_MBA` do cyklu, který je pro horizontální průchod proveden n-krát a pro vertikální n'-krát:

$$n = \left(\frac{width}{8} - 2 \right) \cdot height \,, \tag{7-4}$$

$$n' = \left(\frac{height}{8} - 2 \right) \cdot width \,, \tag{7-5}$$

kde n, n' je počet průchodů funkcí `core_MBA`, $width$ je počet sloupců a $height$ počet řádků v obraze.

Funkce pokračuje rozhodnutím, zdali se jedná o horizontální, či vertikální průchod, podle čeho se načtou příslušné pixely obrazu (`v1` až `v8`). Z hodnot `v3` až `v6` se následně odvodí hodnota dynamiky přechodu `difference`, jejíž výpočet je popsána v kapitole 6.2. Pro omezení nežádoucí filtrace užitečných přechodů je v dalším kroku hodnota `difference` limitována funkcí `gateKeeper` a to v závislosti na průměrné dynamice přechodu blokových artefaktů při dané kvalitě (`ad`). Tímto postupem získáme hodnotu `lim`, kterou dále použijeme pro výpočet váhy analyzovaného přechodu (`balance`).



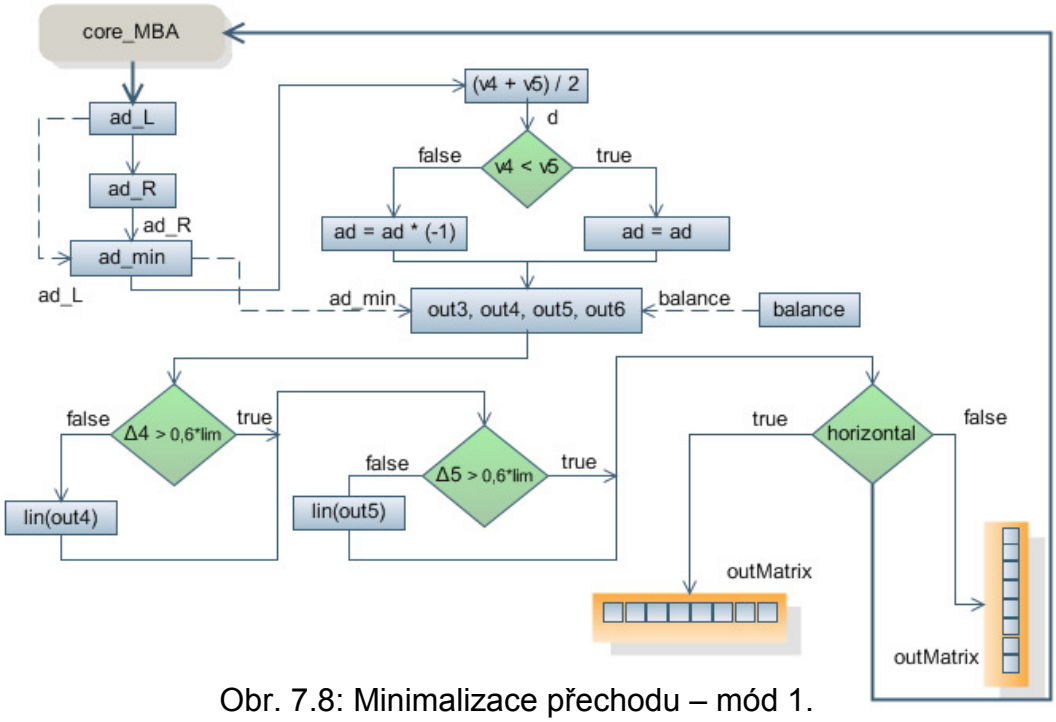
Obr. 7.7: Cyklus funkce `core_MBA`.

Na základě analýzy dynamiky pixelů `v1` až `v8` je rozhodnuto jaký mód filtru bude použit pro samotnou minimalizaci přechodu. Proto je nejdříve vyhledán pixel (vektoru `v1` až `v8`) s nejmenší a nevětší hodnotou, jejíž absolutní rozdíl je testován v podmínce dle rovnice ((6-9)). Jestliže je u oblasti obrazu `v1` až `v8` analyzována nízká dynamika je pro minimalizaci přechodu použit mód 2. V opačném případě je použit mód 1.

7.7.1 Minimalizace přechodu – mód 1

Funkce minimalizace přechodu v módu 1 je spuštěna v případě, že v analyzované oblasti nebyla detekována malá dynamika. Z tohoto poznatku můžeme předpokládat, že se jedná o oblast s vyšším výskytem užitečných detailních informací a proto je pro minimalizaci využita metoda, která zachovává výrazně více těchto detailů oproti filtraci v módu 2.

Myšlenka této metody spočívá ve výpočtu nových hraničních bodů tak, aby byl jejich rozdíl roven průměrnému rozdílu mezi jednotlivými prvky levé, či pravé strany zkoumaného vektoru `v1` až `v8`. Na Obr. 7.8 je zobrazen průběh této metody, jehož postup a výpočty jednotlivých parametrů jsou podrobně rozebrány v kapitole 6.5. Jediným rozdílem je poslední podmínka, která zkoumá, zda-li se jedná o horizontální, či o vertikální průchod. Výsledek této podmínky je určující pro zápis nově vytvořených hodnot do příslušných pozic matice `outMatrix`.

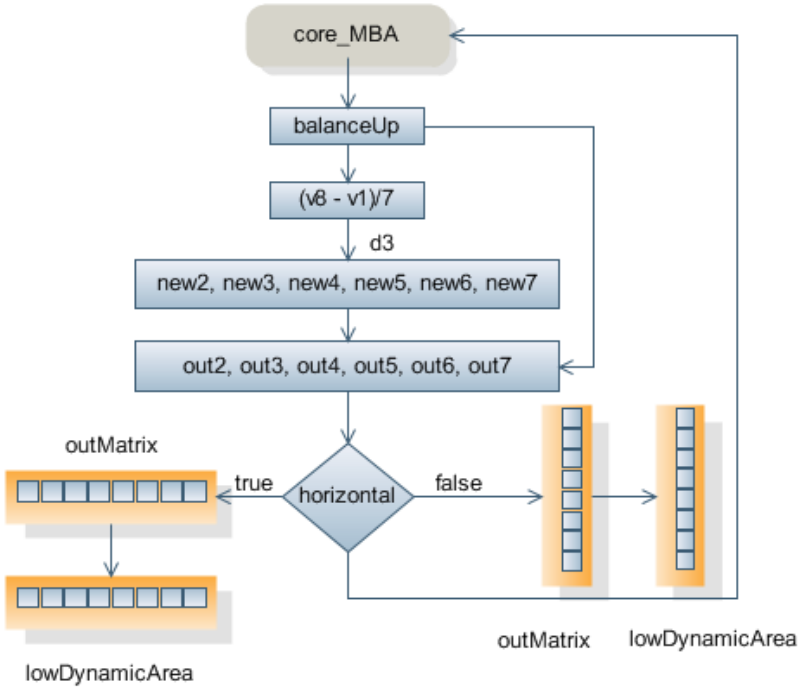


Obr. 7.8: Minimalizace přechodu – mód 1.

7.7.2 *Minimalizace přechodu – mód 2*

V 2. módu, tedy v oblastech s malou dynamikou, je pro minimalizaci nežádoucího přechodu využito silné lineární aproximace, proto je nejprve k hodnotě `balance` přičten offset 55%. V dalším kroku je vypočítána hodnota `d3`, která představuje změnu mezi dvěma sousedními body na lineární přímce mezi `v1` a `v8` (viz rovnice (6-12)). Hodnota `d3` je následně využita pro výpočet nových bodů `new2` až `new7` (viz rovnice (6-13)), které jsou dále transformovány, s použitím hodnoty `balanceUp`, do bodů `out2` až `out7` (viz rovnice (6-14)).

V posledním kroku je rozhodnuto, zda-li se jedná o horizontální, či vertikální průběh, po čemž se následně zapíší vypočítané hodnoty `out2` až `out7` do příslušných pozic ve výstupní matici. Aby bylo možné visuálně ověřit, v jakých oblastech byla použita filtrace pomocí módu 2, jsou navíc zapisovány všechny takto detekované body s hodnotou maximálního jasu (255) do speciálního pole `lowDynamicArea`, které bude následně zpracováno a zobrazeno jako zvláštní obraz, v němž jsou bílou barvou vyznačeny ty oblasti obrazu, ve kterých byla detekována malá dynamika.



Obr. 7.9: Minimalizace přechodu – mód 2.

7.7.3 Převod pole na objekt Image – třída MatrixToImage

Výstupem metody `deblocking_MBA` je pole upravených jasových hodnot, aby je bylo možné zobrazit v některých z grafických komponent, je nutné převést toto pole zpět na objekt typu `Image`. K tomuto účelu byla vytvořena třída `MatrixToImage` obsahující metodu `createImageColorYUV`, která vytváří barevný obraz typu `Image` z upraveného pole jasových složek a z původních neupravených barevných složek.

Funkci metody můžeme rozdělit do dvou částí. Nejprve je nutné převést složky obrazu z barevného prostoru YUV do prostoru RGB, což je provedeno v cyklu s počtem kroků `width`*`height`, kde `width` je počet sloupců a `height` počet řádků. Transformace je tedy provedena pro každý bod obrazu a to dle následujících převodních vzorců:

$$R = Y' + 1{,}140 \cdot V\,, \tag{7-6}$$

$$G = Y' - 0{,}395 \cdot U - 0{,}581 \cdot V\,, \tag{7-7}$$

$$B = Y' + 2{,}032 \cdot U\,, \tag{7-8}$$

kde R,G,B jsou celočíselné hodnoty v rozsahu 0 až 255 vyjadřující intenzitu červené, zelené, modré barvy, Y' je hodnota bodu jasové složky upravená filtrací a U, V jsou hodnoty bodů barevných složek.

V cyklu je dále z vypočítaných jednotlivých složek R, G, B získána pomocí bitového posunu hodnota vyjadřující všechny tři složky a ta je následně uložena do pole `dataOut` délky `width`*`height`. V druhém kroku metody je vytvořen objekt `Image` voláním funkce `createImage` třídy `Toolkit`, jejímiž vstupními parametry jsou rozměry obrazu a pole `dataOut`.

Druhou metodou třídy `MatrixToImage` je `createImageGrey` vracející obraz v odstínech šedi, ve kterém jsou bílou barvou zvýrazněny ty oblasti, u nichž byla detekována malá dynamika. Vstupním parametrem je pole jasové složky `lowDynamicArea`, ve kterém jsou obsaženy pouze hodnoty 255 označující body s malou detekovanou dynamikou. Kdyby byl výsledný obraz vytvořen jen z tohoto pole, obsahoval by pouze bílé bloky a nebylo by příliš patrné, ke kterým částem obrazu konkrétně patří. Proto je druhým vstupním parametrem pole upravených

jasových složek `deblockBrightness` obsahující data z vylepšeného obrazu. Metoda proto v cyklu délky `width*height` nejdříve smíchá obě pole a to tím způsobem, že se pro každý bod testuje `lowDynamicArea` obsahuje-li hodnotu 255. Pokud ano, je tato hodnota uložena na stejnou pozici v poli `deblocBrightness`. Pro ušetření výrazné části výkonu je vytvářen pouze obraz v šedé škále barev, čímž se vyhneme nutnosti přepočtu barevných prostorů. Výslednou reprezentaci bodu v prostoru RGB získáme tak, že použijeme místo jednotlivých barevných složek třikrát složku jasovou.

Vytvoření výsledného objektu `Image` je stejné jako v `createImageColorYUV`, tedy použitím funkce `createImage` třídy `Toolkit`.

8 Testování v DVB-T vysílání

8.1 Simulátory IRT x Osmosys

Tvorba zdrojového kódu aplikace MHP-Deblocking (stejně jako jeho kompilace) probíhala ve vývojovém prostředí Eclipse, které bylo rozšířeno o knihovny MHP 1.0.2. Tento nástroj ovšem defaultně neobsahuje simulátor schopný spustit DVB-J aplikace a proto bylo nutné využít externí aplikace IRT. Tento program simuluje plnohodnotný set-top-box na osobním počítači a jeho prostřednictvím byla vyvíjena větší část aplikace MHP-Deblocking. IRT není pouze simulačním nástrojem, ale je to i jedna z implementací technologie MHP ve skutečných set-top-boxech.

Konkurenční implementací je produkt společnosti Osmosys (tuto implementaci obsahoval i set-top-box, na kterém byla aplikace testována). I zde existuje simulátor pro osobní počítač, který je dokonce integrován do vývojového prostředí Eclipse. Nevýhodou a důvodem proč byl pro vývoj použit především simulátor firmy IRT, je nemožnost spustit Osmosys na libovolném počítači bez použití hardwarového klíče. Vývoj by tak musel být omezen pouze na prostor školní laboratoře, ve které je tento klíč k dispozici, což z časových nároků tvorby aplikace nebylo možné.

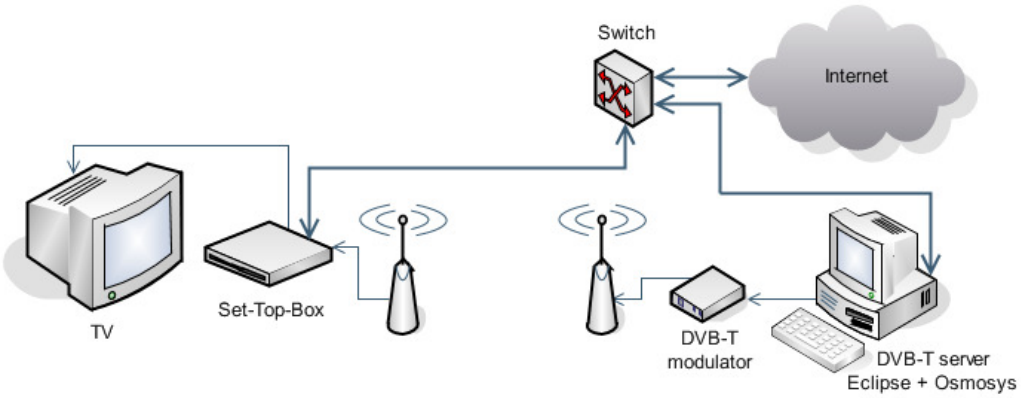
Fakt, že byla pro vývoj aplikace a její skutečný provoz použita rozdílná implementace MHP, by neměl být příliš důležitý. Pokud obě tyto implementace vycházejí ze stejné verze MHP, mělo být chování aplikací na každé z nich totožné. Tento předpoklad není ovšem správný a rozdíly mezi implementacemi IRT x Osmosys jsou v několika případech natolik velké, že ohrožují funkčnost celé aplikace. Níže jsou uvedeny pouze některé z objevených rozdílů:

- rozdíl v grafickém zobrazení komponent – například u komponenty `HToggleButton` není v prostředí Osmosys zobrazována defaultně žádná grafika, proto je nezbytné využít vlastní obrázky popisující aktuální stav checkBoxu,
- `HListGroup` – načítání položek do této komponenty v prostředí IRT probíhá standardně od indexu 0. V prostředí Osmosys je z neznámého důvodu nutné načítat položky od indexu -1, jinak dojde k vyvolání výjimky a k možnému pádu aplikace,
- `HListGroup` – rozdílná identifikace klávesových událostí `keyPreesed` a `keyReleased`,

- `requestFocus` – při zaostření kurzoru na určitou komponentu a následnému překreslení scény, dojde v prostředí Osmosys (na rozdíl od IRT) ke ztrátě tohoto zaostření, čímž je ve většině případů ztracena i možnost dalšího ovládání aplikace,
- `pathSeparator` – v IRT je možné pro oddělení cesty použít znak „/“ stejně jako „\\“, v Osmosys je možné použít pouze znak „/“,
- `File` – v Osmosys je při zadávání jména souboru, prostřednictvím třídy `File`, nutné dbát na velikost písmen, v IRT není `File` Case Sensitive.

Z výše popsaného je tedy patrné, že vzájemná kompatibilita obou implementací je na velmi špatné úrovni. V komerčním provozu je velmi pravděpodobné, že by daná aplikace byla spouštěna jak na set-top-boxech s implementací IRT, tak na set-top-boxech s implementací Osmosys. Proto je při vývoji DVB-J aplikací důležité odladění pro oba systémy.

8.2 DVB-T laboratoř

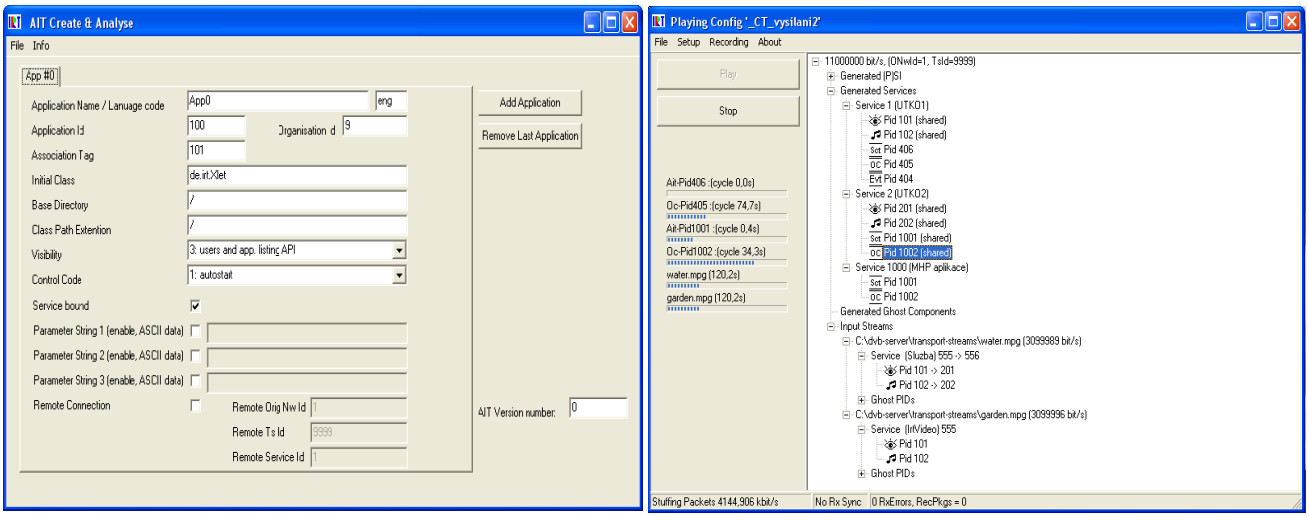


Obr. 8.1: Schéma DVB-T laboratoře.

Schéma pracoviště, které bylo využito pro simulaci, vysílání a testování na skutečném set-top-boxu je na Obr. 8.1. Hlavním prvkem je pracovní stanice vybavená softwarem jak pro psaní a kompilaci zdrojových kódů DVB-J aplikací (Eclipse), tak pro jejich následnou simulaci (Osmosys). Pracovní stanice dále obsahuje DVB-T server, který multiplexuje jednotlivá zdrojová data (video, audio, řídící data, DSM-CC) a generuje z nich datový tok, který je předáván externímu DVB-T modulátoru. Ten tyto data moduluje a vysílá ve vysokofrekvenčním pásmu.

Druhá část pracoviště je tvořena anténou přijímající vysílaný multiplex, který předává set-top-boxu, jenž následně provede demodulaci a zpracování jednotlivých dat. Výstupem set-top-boxu je analogová televize. Námi testovaný set-top-box je dále vybaven zpětným kanálem, který je (stejně jako pracovní stanice) připojen prostřednictvím ethernetového rozhraní k Internetu.

Jak již bylo popsáno, přijímá set-top-box aplikaci MHP-Deblocking prostřednictvím objektového karuselu (DSM-CC). Aby bylo možné přidat aplikaci do vysílání je nutné nejdříve, pomocí programu „AIT Editor“ vytvořit příslušnou AIT tabulku, ve které budou uloženy informace o názvu aplikace, identifikačním kódu, jménu hlavního Xlet souboru a způsobu jakým se aplikace bude spouštět (pressent) viz Obr. 8.2 a). V dalším kroku se vytvořená AIT tabulka předá DVB-T serveru, který data aplikace přidá do multiplexu viz Obr. 8.2 b).



a) **b)**

Obr. 8.2: a) Aplikace pro tvorbu AIT, b) aplikace DVB Playout Server.

8.3 Výsledky testování

Pro testování aplikace ve vysílání byl využit set-top-box Strong 5510 mhp, který je postaven na MHP implementaci firmy Osmosys. Díky dobré vzájemné kompatibilitě mezi simulátorem Osmosys a set-top-boxem, bylo možné provést většinu odladění přímo na pracovní stanici. Nutnost nahrávání kompilovaných dat do multiplexu a jejich následné stahování do set-top-boxu, tak mohlo být omezenou pouze na testování výkonu aplikace.

8.3.1 Výkon set-top-boxu

Z výsledků testování vyplynuly dva klíčové problémy. Tím prvním je výkon set-top-boxu. Již při návrhu filtru MHP-MBA bylo zřejmé, že kritickým parametrem bude právě nízký výkon set-top-boxů. Bohužel zatím neexistují přesnější testy odhalující reálné výkony jednotlivých set-top-boxů, jedinou často užívanou informací v odborné literatuře je, že MHP set-top-boxy jsou v dnešní době osazovány procesory s taktem kolem 200MHz [3]. Proto bylo při vývoji filtru vycházeno z premisy, že reálný výkon skutečného set-top-boxu bude přibližně 10krát až 20krát menší, než výkon osobního počítače s taktem 1,5Ghz.

V Tab. 8-1 jsou zobrazeny časy, které potřebovaly jednotlivé platformy k dokončení deblocking algoritmu. Velmi překvapivé hodnoty byly naměřeny na set-top-boxu (Strong 5510 mhp), kterému trvalo dokončení algoritmu více jak 37 sekund. Takto vysoká hodnota by mohla znamenat příliš velkou výkonovou náročnost samotného filtru, ovšem pokud ji porovnáme s dobou zpracování na osobním počítači (v prostředí Java SE) zjistíme, že je více jak 1681krát delší. Z dalších naměřených výsledků (viz Tab. 8-2) vyplývá, že zpracování deblocking algoritmu trvá pouze 30% času. Zbylých 70% zabírá:

- převod obrazu z objektu třídy `Image` do formy pole RGB (prostřednictvím metody `PixelGrabber`),
- konverze barevného prostoru z RGB do YUV a naopak,
- převod pole RGB zpět do formy objektu `Image`.

Tab. 8-1: Doba zpracování deblocking algoritmu.

	MHP-MBA [ms]	**H.263 [ms]**	**Δ[%]**[6]
JavaSE	22	15	46,6
MHP-IRT	47	31	51,6
MHP-Osmosys	380	238	59,7
Strong 5510	37 000	19 000	94,7

[6] Udává o kolik procent je pomalejší filtr MHP-MBA oproti filtru z H.263.

Celková doba zpracování obrazu o velikosti 512x512 bodů je tedy přibližně 120 sekund, což je pro reálný provoz nepoužitelné. Je ovšem důležité podotknout, že aplikace byla testována pouze na set-top-boxu Strong 5510 mhp, který patří do nejnižší cenové kategorie set-top-boxů vybavených technologií MHP. Je proto více než pravděpodobné, že jsou již dnes na trhu dostupné MHP set-top-boxy s výrazně vyšším výkonem [9].

Tab. 8-2: Celková doba zpracování včetně konverze obrazů.

	Celkové zpracování (MHP-MBA) [ms]	**Konverze obrazu [ms]**	**Δ[%]** [6]
JavaSE	75	53	70,6
MHP-IRT	220	173	78,6
MHP-Osmosys	1332	952	71,4
Strong 5510	119 800	82 800	69,1

8.3.2 Zobrazení přirozených obrazů

Aplikace MHP-Deblocking je určena především pro vylepšování přirozených obrazů ve formátu JPG. Tyto obrazy mají obecně hloubku barev 24bitů, což je více jak 16 miliónů barev. Ze standardu MHP ovšem vyplývá výrazné omezení v možnosti zobrazení přirozených obrazů.

Výsledný televizní obraz je v MHP složen minimálně ze tří vrstev:

- „background plane“ – v každém set-top-boxu musí tato vrstva podporovat zobrazení přirozeného obrazu ve formátu MPEG2 I-frame v rozlišení 720x576 bodů,
- „video plane“ – umožňuje zobrazit formát „MHP Video Dripfeed“ (přirozené video ve formátu MPEG2 I-Frame nebo P-Frame),
- „graphics plane“ – tato vrstva musí podporovat zobrazení grafiky minimálně se 139 netransparentními barvami.

Pokud vykreslujeme jakoukoli grafiku (tlačítka, obrazy) prostřednictvím standardních grafických komponent, vykreslujeme ji do vrstvy „graphics plane“. Zde

[7] Udává jakou procentuální část, z celkové doby zpracování, trvá konverze obrazu.

nastává problém při potřebě vykreslení přirozeného obrazu. Ten bude totiž automaticky konvertován z 24bitové hloubky do 139 barev, což způsobí výraznou degradaci obrazu.

Tento případ nastal i na testovaném set-top-boxu Strong 5510 mhp (viz Obr. 8.3). Jedinou možností jak zobrazit přirozený obraz, bez degradace hloubky barev, je využití vrstvy „background plane“. Ta sice umožňuje zobrazení v barevné hloubce 24bitů, ovšem pouze ve formátu MPEG2 I-Frame (mpg, mv2). To by nebylo překážkou v případě, kdy bychom potřebovali pouze zobrazit soubory stáhnuté přes objektový karusel (DSM-CC), či zpětný kanál. V takovém případě, by bylo možné provést potřebnou konverzi formátu (JPG do MPEG2 I-Frame), ještě před přidáním obrazu do objektového karuselu. My ovšem potřebujeme vylepšovaný obraz editovat, což by v důsledku znamenalo provádět konverzi formátů přímo v set-top-boxu. K čemuž ovšem není dostupné jak výkonové, tak programové vybavení. Další překážkou ve využití vrstvy „background plane“ je nutnost konstantního rozlišení zobrazovaného obrazu 720x576 bodů.

Obrazy v aplikaci MHP-Deblocking mají tak i nadále degradovanou hloubku barev[8]. Tento problém není ovšem způsoben samotnou aplikací, ale platformou MHP, která dovoluje podporu pouze 139 barev na grafické vrstvě. Tento stav je pouze přechodný a již dnes je na trhu řada set-top-boxů podporujících na grafické vrstvě plnou 24bitovou hloubku barev [3].

Obr. 8.3: Přirozený obraz s degradovanou hloubkou barev.

[8] na set-top-boxu Strong 5510 mhp

9 Závěr

Cílem této práce bylo vytvořit algoritmus minimalizující blokové artefakty způsobené DCT kompresí a to v prostředí pozemního digitálního televizního vysílání. Dílčím výsledkem práce je proto adaptivní MHP-MBA filtr 1.0 účinně odstraňující právě tyto blokové artefakty. Pro použití v DVB-T byla vyvinuta DVB-J aplikace MHP-Deblocking, která v sobě implementuje jak deblocking filtr MHP-MBA, tak filtr z kodeku H.263.

Největší podíl na výstupních parametrech filtru, nebo-li úspěšnosti s jakou dokáže minimalizovat rušivé artefakty, má jeho adaptivita. Tu u realizovaného filtru můžeme rozdělit do tří částí. Základní je funkce detekce přechodů, která má za úkol vyhledat a určit úroveň „falešných“ rušivých přechodů. Tato schopnost je důležitá především z pohledu zachování detailních užitečných informací vylepšovaného obrazu. Dalším stupněm implementované adaptivity je mód hladkých oblastí, který má za úkol vyhledat části obrazu s malou dynamikou. V těchto oblastech je následně použita výrazně silnější metoda filtrace, která vykazuje především ze subjektivního hodnocení, viditelně vyšší kvalitu vnímání vylepšeného obrazu. Třetí zcela zásadní adaptivitou je řízení celého filtru na základě znalosti kvality komprese použité u vylepšovaného souboru.

Kvalita a parametry realizovaného filtru byly porovnávány především s deblocking filtrem video kodeku H.263. Z naměřených výsledků vyplývá, že MHP-MBA filter 1.0 má jak z objektivního, tak subjektivního hlediska vyšší kvalitu výstupního obrazu, což je umožněno především díky více módové adaptivitě. Oproti H.263 je nevýhodou celkově větší složitost filtru, která si vybírá daň v podobě nárůstu doby zpracování v rozsahu 45 až 95% (v závislosti na použité platformě).

Při testování aplikace ve skutečném DVB-T vysílání (na set-top-boxu Strong 5510 mhp) byly zjištěny dva zásadní problémy. Tím prvním byl nedostatečný výkon set-top-boxu, který zpracovával obraz o rozměrech 512x512 bodů téměř 120 sekund, což je v porovnání se zpracováním na osobním počítači (na platformě JavaSE) více jak 1590krát déle. Druhým problémem bylo zobrazení přirozených obrazů na grafické vrstvě set-top-boxu. Dle specifikace MHP může tato vrstva podporovat i 24bitovou hloubku barev, minimálně však musí každý set-top-box zobrazit pouze 139 netransparentních barev. To je i případ set-top-boxu Strong 5510 mhp, který tak vlivem automatické konverze (z 24bitové hloubky do 139 barev) způsobuje výraznou degradaci výsledného obrazu.

Oba popsané problémy nejsou způsobeny platformou MHP, ale jsou závislé výhradně na hardwarovém vybavení konkrétního set-top-boxu. Námi testovaný Strong 5510 mhp, patří mezi nejlevnější MHP set-top-boxy na našem trhu, čemuž také odpovídá jeho výkon. V současné době se na trhu objevují set-top-boxy, které by podle techických specifikací měly splňovat požadovaná kritéria pro plnohodnotnou funkci aplikace MHP-Deblocking. Tyto set-top-boxy mají dostatečný výkon a schopnost zobrazení přirozených obrazu bez degradace hloubky barev.

Použitá literatura

[1] Morris, S., Chaigneau, A. *Interactive TV standards*. 2005. ISBN 0-240-80666-2

[2] Říčný, V., Kratochvíl, T. *Základy televizní techniky*. Skriptum ÚREL FEKT VUT v Brně, 2002.

[3] MORRIS, S. *The MHP Tutorial.* Dostupné z WWW: http://www.interactivetvweb.org/tutorial/mhp/index.shtml.htm (listopad 2007)

[4] MORRIS, S. *The JavaTV Tutorial*. Dostupné z WWW: http://www.interactivetvweb.org/tutorial/javatv/index.shtml.htm (listopad 2007)

[5] EVAIN, J.-P. *The Multimedia Home Platform*. Dostupné z WWW: http://www.dvb.org\documents\white-papers\evain.pdf (říjen 2006)

[6] LIŠKA, D. *Vznik, současnost a budoucnost DVB*. Dostupné z WWW: http://www.digitaltv.cz (říjen 2005)

[7] ZITA, A. *Experimentální vysílání DVB-T*. Dostupné z WWW: http://www.cra.cz (duben 2005)

[8] DVB. *Multimedia Home Platform (MHP) Specification 1.1.2.* Dostupné z WWW: http://www.mhp.org\mhp_technology\mhp_1_1\mhp_a0068r1.pdf (září 2007)

[9] Legíň, Martin. *Televizní technika DVB-T*. 1. vyd. PRAHA: BEN, 2006. 288 s. ISBN 80–7300-204–3.

[10] BODEČEK, Kamil. Post-processing JPEG obrazu v interaktivních. *Elektrorevue* [online]. 2007 [cit. 2007–10-12], s. 29/1-29/18. Dostupný z WWW: <http://www.elektrorevue.cz/cz/download/post-processing-jpeg-obrazu-v-interaktivnich-aplikacich-mhp/>. ISSN 1213–1539.

[11] HSU, Yuh-Feng, CHEN, Yung-Chang. A NEW ADAPTIVE SEPARABLE MEDIAN FILTER FOR REMOVING BLOCKING EFFECTS. *IEEE Transactions on Consumer Electronics* [online]. 1993 [cit. 2007–11-09], s. 510–514. Dostupný z WWW: http://ieeexplore.ieee.org/url=/iel4/1037/7500/00697631.pdf.

[12] PARK, Hyun, LEE, Yung. A Postprocessing Method for Reducing Quantization. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*. 1999, no. 1, s. 161-171.

[13] SUTHAHARAN, Shan, WU, H, YUEN, Michael. A New Post-Filtering Technique for Block-Based Transform Coded Images.. *IEEE TENCON*. 1996, no. 1, s. 702-705.

[14] ALTER, Francois, DURAND, Sylvain, FROMENT, Jacques. DEBLOCKING DCT-BASED COMPRESSED IMAGES WITH WEIGHTED TOTAL. *ICASSP*. 2004, č. 1, s. 221-224.

[15] TIŠNOVSKÝ, Pavel. Načtení informací ze souborů typu JFIF/JPEG. *ROOT.CZ* [online]. 2007 [cit. 2008-10-2]. Dostupný z WWW: http://www.root.cz/clanky/programujeme-jpeg-nacteni-informaci-ze-souboru-typu-jfifjpeg/.

PŘÍLOHY

Příloha 1: CD s PDF verzí této práce a zdrojovými kódy aplikace MHP-Deblocking.